University of Nevada, Reno

RAIN and NCS 5 Benchmarks

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science

by

Milind A. Zirpe

Dr. Frederick C. Harris, Jr., Thesis Advisor

December, 2007

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

MILIND A. ZIRPE

entitled

RAIN and NCS 5 Benchmarks

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Frederick C. Harris, Jr., Ph.D., Advisor

Bobby D. Bryant, Ph.D., Committee Member

Philip H. Goodman, M.D., Graduate School Representative

Marsha H. Read, Ph. D., Associate Dean, Graduate School

December, 2007

Abstract

The primary objective of the Brain Computation lab at University of Nevada, Reno is to discover principles and develop models of social intelligence in an artificial agent, with biology as basis. To help with this aim, a complex and relatively biologically realistic spiking neural network simulator was developed. This is the NeoCortical Simulator version 5 (NCS 5) which is capable of efficiently simulating large neural networks (more than 10,000 cells and 1,000,000 synapses) using a parallel cluster. The work done in this thesis develops neural network models which exhibit the principle of background activity present in a live biological brain using NCS. The principle of Recurrent Asynchronous Irregular Network (RAIN) might provide a basis for developing more advanced human aspects of memory, learning, consciousness, and pattern recognition and various other application fields. Furthermore, benchmarks were done to test neural networks used in a Virtual Social Robot (VSR) loop. Results of these benchmarks showed capabilities of our cluster and current software which would prove vital for future upgrades and design of neural network models using NCS.

Dedication

This work is dedicated to my parents: Arvind and Latika. I would not be here without you and your help. I am eternally thankful to you.

Acknowledgments

Thanks to Dr. Philip H. Goodman for giving this marvelous opportunity to work and be a part of the Brain Lab and being patient. He has been a lighthouse in a dark night helping me in all aspects of my work at the lab.

Thanks to Dr. Frederick C. Harris, Jr. for accepting to be my advisor and showing a lot of patience. His guidance and suggestions have been invaluable to me.

Thanks go out to Dr. Bobby D. Bryant who agreed to be on my committee in a time of need.

I would also like to thank, James King and James Frye for helping me out with the NCS 5 code, Rich Drewes for giving technical advice and help with NCS 5, John Kenyon for managing Cortex, Sermsak Buntha, Dr. Quan Zou and Dr. Monica Nicolescu.

I would also like to thank Office of Naval Research for their support of Brain Computation Lab (ONR Grant # N00014-07-1-0018).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Nervous system of a living organism is by far the most complex system. This is very true in case of mammals and especially humans. Even today, we are in the process of understanding the detailed anatomy of the Human Nervous system and are far from completion. But the real major challenge is to learn how it actually functions. The Brain is the core of a Nervous system with more than 100 billion neurons in an adult human [63, 71] and about 10 to 50 times more glial cells [6]. There have been a lot of discoveries and breakthroughs in this regard, yet we still know so little about the functioning of this magnificent organ.

The term Artificial Intelligence (AI) was coined in 1956 by John McCarthy [25, 62]. Artificial Intelligence is the study and design of intelligent agents, where an intelligent agent is a system that perceives its environment and takes actions which maximizes its chances of success [59]. But after a few decades more and more people are realizing that AI has a lot of limitations and we possibly cannot hope to create a truly human-like intelligent agent anytime soon in the future.

As an alternative to AI, there has been a biological modeling approach to this problem. In this approach, the biological nervous system is simulated with the help of computers and simulator software to produce biologically realistic spiking neural network simulations. NEURON [13] and GENESIS (The GEneral NEural SImulation System) [9, 22] are among the first simulation environments. They offer single neuron and small

realistic neuron network and synapses simulation abilities. The Brain Computation Lab at the University of Nevada, Reno, under the direction of Dr. Philip Goodman, undertook a project for designing relatively realistic biological neural networks which would allow simulation of large networks (more than 10,000 cells or neurons and 1,000,000 synapses) quite efficiently [10]. The result was a simulator program called NCS (NeoCortical Simulator) [12].

This thesis researches the principle of simulation of background activity in a biological brain using NCS version 5 (NCS 5). In addition, a series of benchmark tests were performed to evaluate the performance of NCS 5 (the latest version of NeoCortical Simulator) on our cluster [7]. A series of other benchmarks, based on the benchmarks in this thesis, have been planned for other hardware systems.

The rest of this thesis is structured as follows. In Chapter 2, we review some basic biological concepts in brief. They are necessary to understand NCS 5 neural models written in "input" (*.in) files. Then we will review NCS itself by taking a look at its history, structure of the input file, usage, and applications. We will then take a brief look at neural modeling. And finally we'll have our first look at the concept of Recurrent Asynchronous Irregular Network (RAIN). Chapter 3 explains how RAIN is actually created, from initial neural model to its evolved version conducive for RAIN activity. In Chapter 4, we will take a look at benchmark tests done for NCS 5 and Brainstem [58]. System setup, Procedure and Tests Design are explained. Results are discussed in Chapter 5. RAIN and Benchmark tests results are discussed here. Finally, Chapter 6 discusses conclusions and interesting future work based on RAIN.

Chapter 2

Background

We need to familiarize ourselves with biology that NCS (NeoCortical Simulator) has modeled. This chapter presents a review of basic biological concepts from the perspective of modeling mammalian brain networks, information about the simulator NCS, and neural modeling.

2.1 Brief Overview of Biological Concepts

A mammalian brain is composed of several different types of complex structures. Our structure of interest is the cortex, where cognition and related functions takes place [31, 40]. Essentially, the cortex is a highly convoluted thin sheet of large number of cells (neurons) and their various associated structures. The cortex sheet is quite large compared to the space available in brain. Nature has found a means of fitting cortex by folding it in three dimensions, which helps in conserving space and also reduces inter-cellular distances leading to efficient use of neural wiring material, required to form communication connections among neurons.

Neurons

The brain consists of a basic type of cell called as neuron. The number of neurons in an adult human brain is very large (more than 10^{11}) and there are various types [63, 71]. Another type of cell, called glial cells are also present. There are various types of glial

cells. Neuroglia cells do not conduct nerve impulses, but instead, they support, nourish, and protect the neurons. They are far more numerous than neurons [6] and, unlike neurons, are capable of mitosis. Neurons, or nerve cells, carry out the functions of the nervous system by conducting information via nerve impulses. They are highly specialized and amitotic [60]. This means that if a neuron is destroyed, it cannot be replaced because neurons do not go through mitosis. Figure 2.1 shows a typical neuron structure.

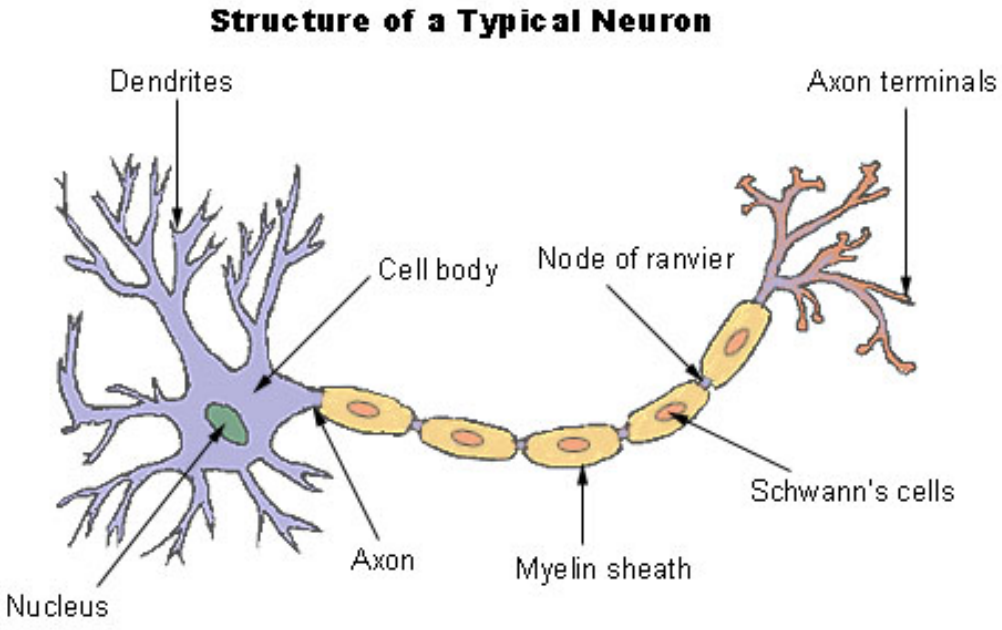


Figure 2.1 A typical Neuron. [3]

A typical neuron consists of a cell body (soma) with a nucleus and numerous structures called dendrites which expand from the soma. Surfaces of dendrites primarily exchange chemical information with other neurons via synapses. An axon is a long dendrite different from all others. The purpose of an axon is to transfer electro-chemical signals to other neurons, sometimes over a long distance. The axon is protected by a

myelin sheath, which helps in electro-chemical signal transmission. The exchange of information (essentially a spike) to other neurons happens between the end of axon (which is shown as axon terminal) and dendrites with help of synapses [60].

Action Potential

Figure 2.2 gives a schematic representation of action potential.

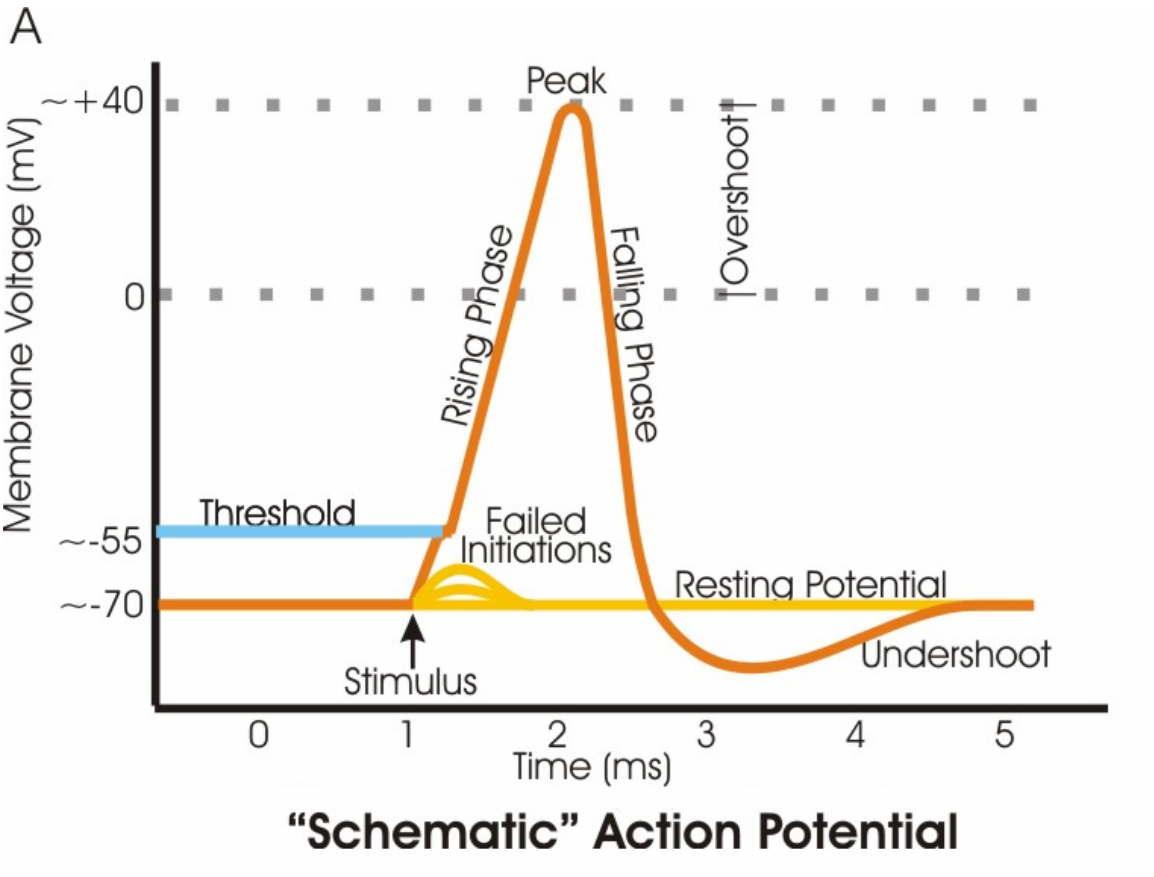


Figure 2.2 Action Potential. [4]

Each neuron is connected by axons, dendrites and synapses to a number of other neurons, typically about a thousand. Neurons maintain a small electrical potential, normally about -70 mV in their soma, due to the interactions of channels in their cell walls with ions in the intercellular fluid [60]. This voltage continually changes in response to external inputs, primarily stimuli received from incoming synapses. When the

voltage reaches a critical threshold (spiking threshold), these ion channels cause an abrupt rise and fall in the cell voltage. This is called an action potential (AP) or "a spike firing."

Synapse

Each of the 100 billion neurons has on average 7,000 synaptic connections to other neurons. It has been estimated that the brain of a three-year-old child has about 10^{16} synapses (about 10 quadrillion). This number declines with age, stabilizing by adulthood. Estimates vary for an adult, ranging from 10^{15} to 5 x 10^{15} synapses (which is 1 to 5 quadrillion) [28]. Figure 2.3 shows a synapse.

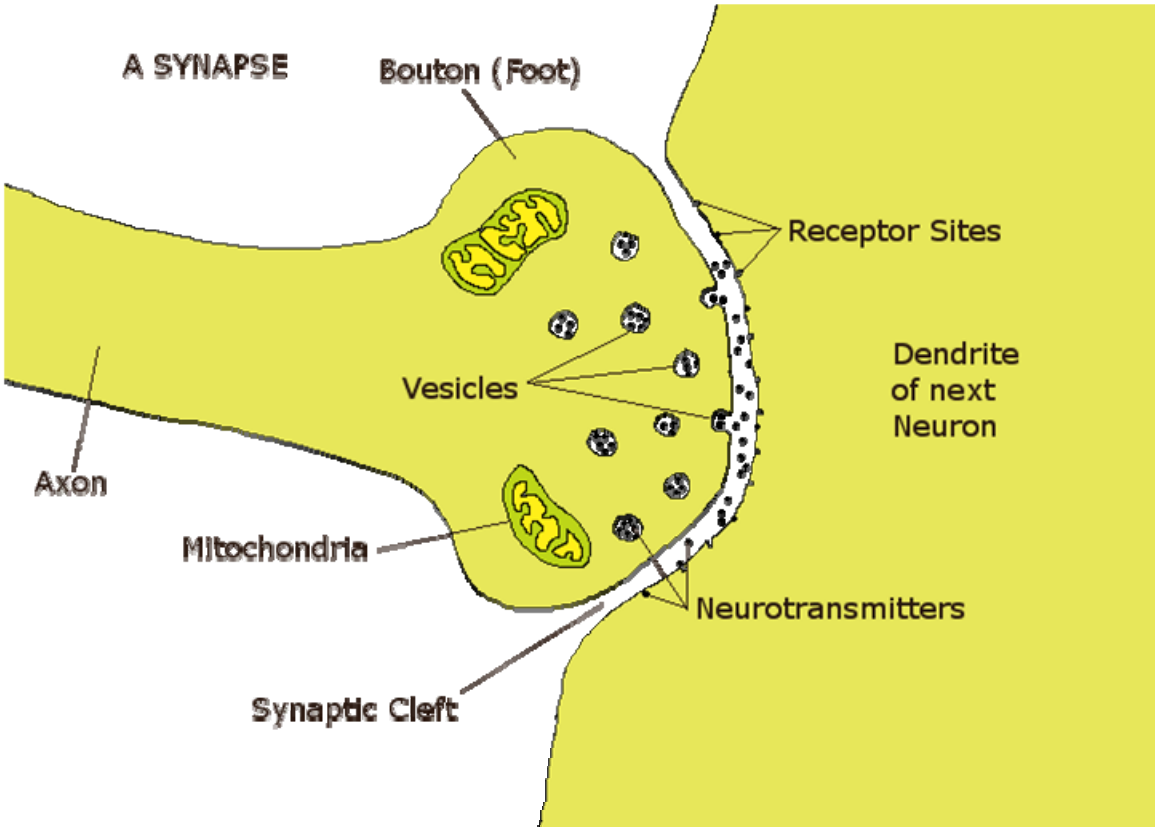


Figure 2.3 A Synapse. [2]

As seen from the figure, a synapse mainly consists of three structures:

1. Axon terminal of a pre-synaptic neuron which has vesicles containing neurotransmitters, mitochondria, and other cell organelles.
2. A synaptic cleft or small tiny gap between pre- and post-synaptic neurons.
3. A dendrite of a post-synaptic neuron which has receptor sites for accepting the neurotransmitters.

As mentioned before, information is passed from one neuron to another in form of a spike which is caused by an action potential. A synapse is the structure which actually makes this possible. A synapse is a very tiny gap between the axon ending of one neuron and the dendrite of the next neuron [44]. The neuron which sends information is called pre-synaptic neuron and the receiving neuron is called the post-synaptic neuron.

When the action potential reaches the axon ending, it causes tiny bubbles of chemicals called vesicles to release their contents into the synaptic gap [60]. These chemicals are called neurotransmitters. These sail across the gap to the next neuron, where they find special places on the cell membrane of the next neuron called receptor sites. The neurotransmitter acts like a little key, and the receptor site like a little lock. When they meet, they open a passage way for ions, which then change the balance of ions on the outside and the inside of the next neuron. When the spike pulse reaches the destination cell, it combines with the inputs from all the cell's other incoming synapses, which in turn may cause the destination cell to fire and the whole process starts over again.

Some types of synapses change their behavior over a period of time, in response to these action potential inputs. This process has been defined as Hebbian Learning [42]. In positive learning, a neuron's response to incoming spikes is increased gradually

depending upon the timing and number of spikes received in past. On the other hand, if a neuron's response decreases gradually after receiving spikes, it is called negative learning.

Columns and Layers

Conceptually the human cortex is organized horizontally into columns [56], which are localized regions of high neural connectivity. From the size of the cortex and the typical size of a column, it can be estimated that there are about 2×10^6 columns in humans [43]. There may be more if the columns can overlap [66]. The cortex is further divided vertically into layers that are distinguished by variations in the types of neurons. Most of the cortex has six of these layers; the hippocampus has only four. It should be noted that columns and layers are not in any sense separate organs; they are distinguished anatomically only as patterns of cell distribution and connection. In context of NCS, they are used in the input file constructs that define a particular brain model and for organizational purposes. By themselves they have no function (i.e. behavior) that can be modeled.

2.2 The NeoCortical Simulator (NCS)

The NeoCortical Simulator (NCS) has been developed at the Brain Computation Laboratory (BCL) [10], under the direction of Dr. Philip Goodman, at the University of Nevada, Reno (UNR). Over a period of about 10 years, NCS has undergone many transformations and revisions. This section details those changes. We will also study the

structure of the input file and how to use NCS. At the end of this section, some applications of NCS used in experiments and projects are discussed.

2.2.1 History and Overview of NCS

Over the years research has been going on utilizing AI approaches for the generation of a socially intelligent entity. Nature served as an inspiration for some of these approaches to reach to try to achieve an efficient solution [36, 37, 38]. Current technology has limitations for using actual organic material for such architecture. The next best alternative is to use silicon-based computational devices (viz. computers). Thus a software simulator was created, which would model biologically realistic large spiking model of cell networks mimicking the activity present in cortex. The simulator would be able to model the neo-cortex, the hippocampus, and other structures with a fair degree of complexity. This simulator was the NeoCortical Simulator.

Humans have a large amount of the cortex termed “neo-cortex” (new cortex) which is lacking in other mammals (and animals). Debate is going on whether this neo-cortex is indeed the key to the intelligence in humans. Hence the simulator was named as NeoCortical Simulator. NCS was made accessible to the outside world in 2000.

The First Two Versions of NCS

Development of the first version of NCS began in 1997 by Philip Goodman in collaboration with Henry Markram, currently the co-director of “Brain Mind Institute” at École Polytechnique Fédérale de Lausanne (EPFL) in Lausanne, Switzerland, and Thomas McKenna (Neural Computation Program Officer at the U.S. Office of Naval Research) with MATLAB [41, 51] as the programming environment [24]. At the same

time there were important discoveries made with regards to synaptic plasticity and connectivity [53, 54] which helped in the design of NCS.

This first version of NCS (NCS version 1) used a template of Action Potential (AP) spike waveform (as shape of these waveforms have less variation) which was efficient. Cell membrane dynamics and calcium dependent AHP channels, voltage dependent A- & M- channels (ion channel dynamics) were also modeled to compute a spiking network along with implementation of synaptic delay [29, 45]. Using published results from Henry Markram and others [65], changes in short-term synaptic dynamics and long-term Hebbian regulation of synaptic efficacy were also added.

Initial promising results led to funding by Office of Naval Research (ONR) in 1999 [38]. This helped in collaboration with other professors from UNR. The MATLAB version of NCS was slow and serial (i.e. it could not be distributed to run on multiple nodes in parallel), limiting the number of cells and synapses that could be simulated in a network. Due to this a need for a more efficient simulator was felt. This led to the development of the first C version of NCS.

In 1999, Ali Etezadi-Amoli programmed the C version of NCS – NCS version 2 [29]. This was a direct implementation of NCS 1 with no major additional features. Gain in performance was about an order of magnitude over NCS 1. Another student, Keith Wesolowski made improvements to the code and added parallelism to NCS 2. Experiments were carried out to test the performance of NCS 2 on multiple CPUs. A 20 node Beowulf cluster (450 MHz Pentium CPUs) was used. During experimentation, size of the network was increased from 100 cells to 1000 cells and number of CPUs was

increased from 1 to 10 CPUs. Further increase in CPUs did not provide a marked gain in runtime of NCS.

Figure 2.4 shows the performance result of increasing the number of nodes for a 1000 neuron simulation. The large difference between ideal and actual results was due to slow switching across Ethernet. A Myrinet switch and faster CPU nodes were added later on. Further addition of features followed.

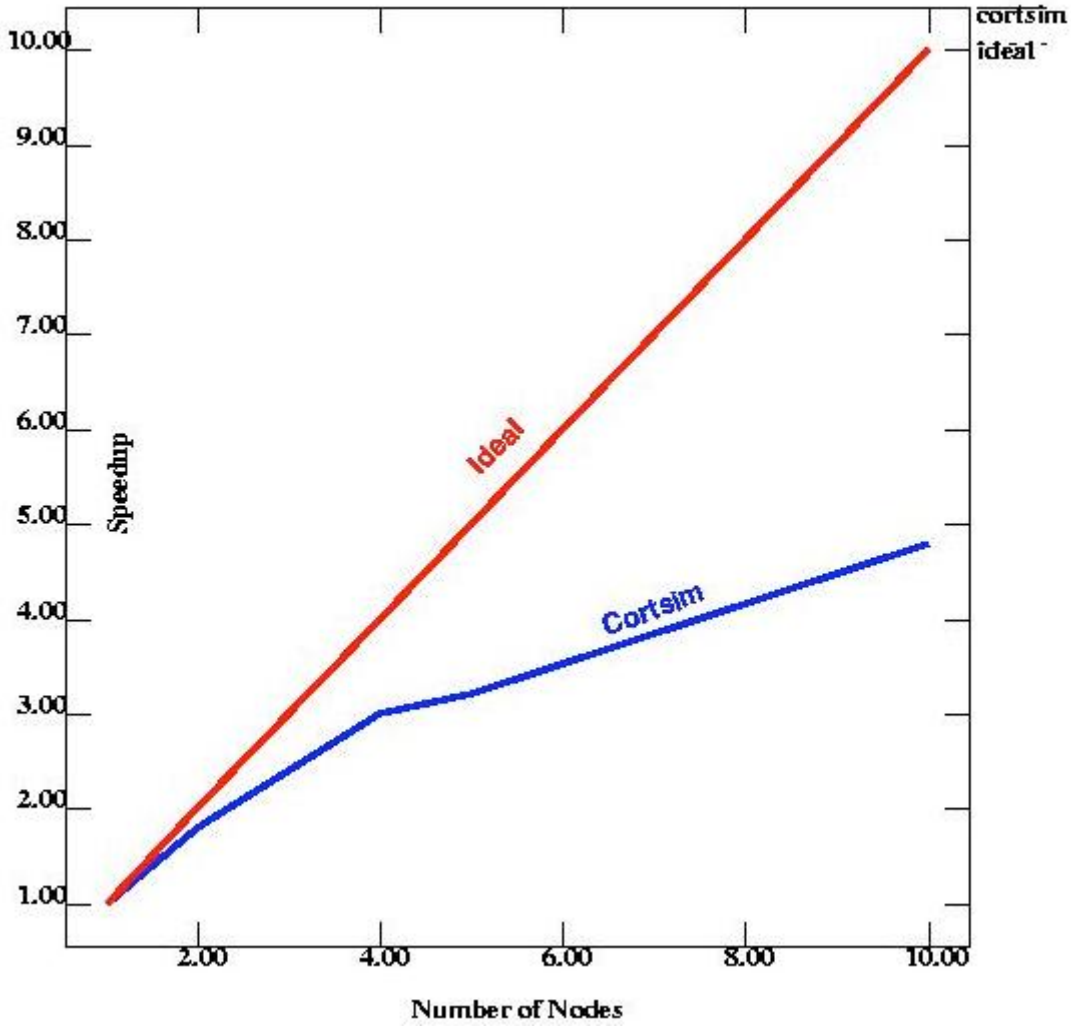


Figure 2.4 Performance result of a parallelized NCS 2. [38, 45]

NCS Version 3

In 2000, there was a major overhaul of the NCS code [72]. This was the first C++ version of NCS – NCS version 3 and aimed to utilize object-oriented design to improve code performance and organization [74, 73]. Relationships between neuronal properties were broken down into corresponding objects. This accomplished two objectives. Since objects were fully encapsulated, different implementations could be swapped in and out without the need of changing large portions of code. This allowed quick mathematical and algorithmic changes to NCS code. The other objective was to have better code organization by abstracting various biological concepts. It was made sure that this version of NCS was generalized for future improvements and capable of running more efficiently on a parallel system.

NCS 3 used MPI standard parallel libraries for distributing an NCS simulation over a cluster of CPUs. MPI is still used in NCS 5, for communication of spike messages between cell groups via the Message Bus. Cells were split up equally across the nodes with no intention of keeping cell groups together, although load-balancing algorithms were anticipated to be included in the future [45]. A Message Bus object was used to co-ordinate communication flow between objects on and off a node. Timing and synchronization of simulations to prevent deadlock and starvation were thus handled. Figure 2.5 shows the major components of NCS 3 interacting with Message Bus object.

New hardware was acquired to increase performance and network simulation size [36]. This was the Cortex cluster consisting of 30 dual Pentium III (1 GHz) processor nodes with 4 GB installed RAM per node. A Myrinet switch was also installed for faster network message packet switching. Introduction of a Myrinet switch improved

simulation performance a great deal. Currently, Cortex has 200 CPUs consisting of Pentium and AMD processors.

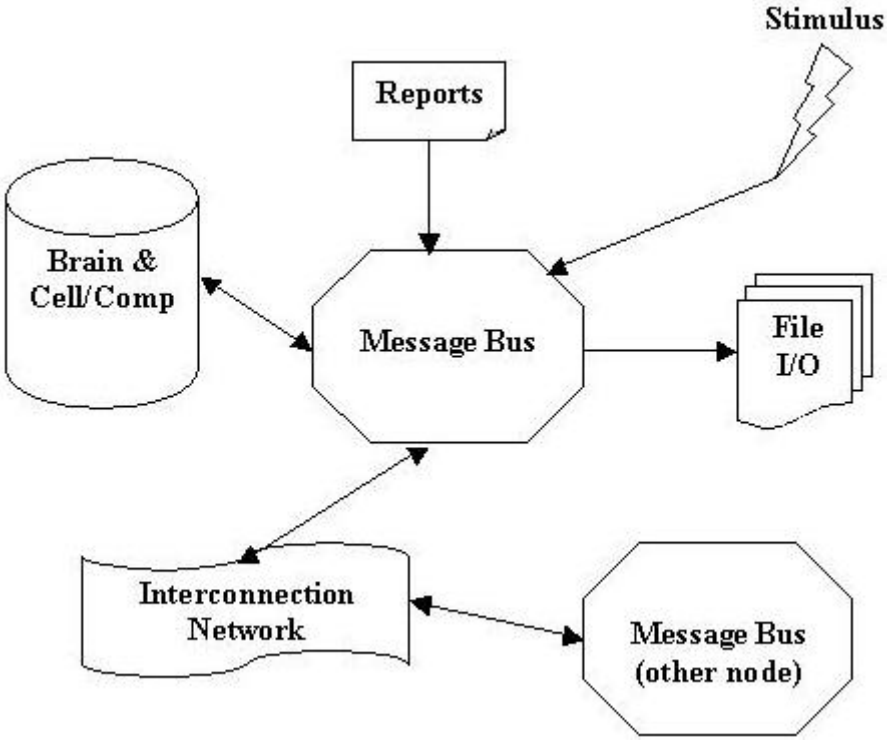


Figure 2.5 Communication scheme of NCS 3 using Message Bus object. [72]

Experiments were performed using NCS 3, to use channel structures within cellular compartments so as to reflect accurate dynamics of the inter-neuronal GABAergic systems [50]. Previously published *in vitro* responses taken after applying one second current steps from 50 to 350 pA were replicated by combining channels properly. Figure 2.6 gives the results by NCS 3 in replicating the behavior of various neurons: classic non-accommodating (cNAC), classic accommodating (cAC), bursting non-accommodating (bNAC), bursting accommodating (bAC), delayed non-accommodating (dNAC), delayed accommodating (dAC), classic stuttering (cSTUT), and bursting stuttering (bSTUT) [50].

Except for “stuttering” types, the behaviors were relatively robust under a two-fold variation in strength for a single somatic cell [45].

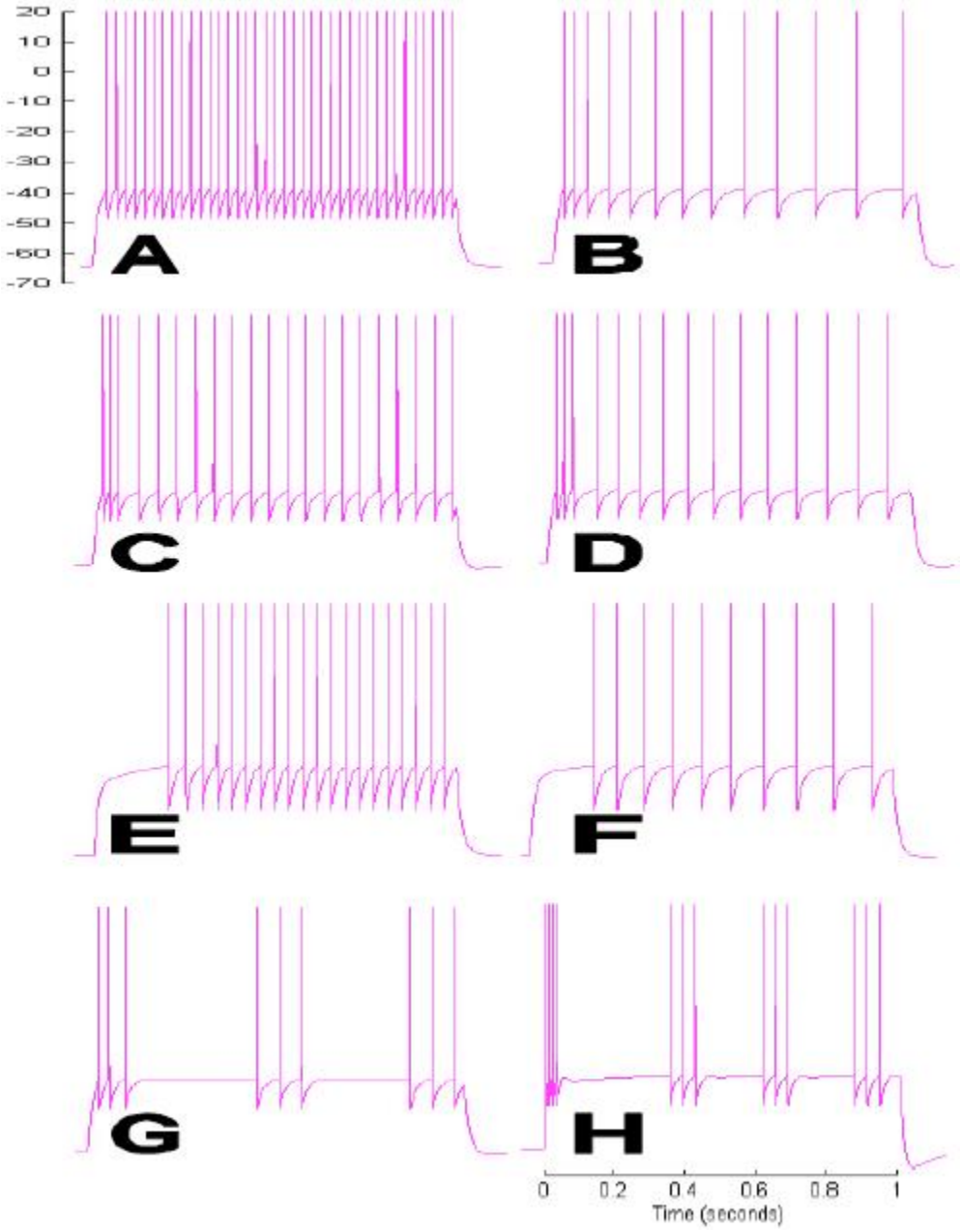


Figure 2.6 Simulated voltage response to step current from 150 to 300 pA. [50]

(A) cNAC (B) cAC (C) bNAC (D) bAC (E) dNAC (F) dAC (G) cSTUT (H) bSTUT

NCS Version 4 and NCS Version 5

It was found that NCS 3 still had some scope for improvement. C language seemed to be more efficient for some functionality than C++. Thus, there was another change to optimize NCS, this time, almost all the C++ object oriented features were replaced by a more compact C code. Simulation functionality was expanded and further speed and performance optimizations were done [37].

These changes were done during 2001 [32, 29]. Those efforts lead to a new release of NCS 3 in 2003. Subsequently NCS version 4 (in 2003-2004) and NCS version 5 (later in 2004) followed with elimination of errors and bugs from the code and further optimization.

Previous NCS versions worked with point neurons (i.e. only a soma and a compartment) only. The new revisions of NCS permitted complex cells consisting of numerous compartmental structures (dendrites, spines and axons) to be created. The neuron model could also be designed and placed in a three dimensional space. For example, synapses present along closely spaced objects had a higher probability for forming connections. Similarly, action potentials would travel faster to closer destinations. The state of a brain (neural network) could now be saved and loaded back. This helped to preserve and reuse (for future simulations) synaptic dynamics that were induced by Hebbian learning during an earlier simulation.

Many more optimizations were made to NCS. Load balancing in NCS 3 used number of cells as a decisive factor in even distribution of the network over the cluster. There was a risk that some processors in the cluster were being overworked while others stayed idle, creating a need for improved load balancing techniques [32, 33]. The newer methods also

considered the number of synapses along with number of cells during distribution step of simulation setup. This made sense, as synapses were involved in message passing and had dynamics of their own which needed to be computed at each simulation time tick. Synapses also required a large share of the memory needed for brain simulation.

The connection process was optimized, reducing time taken in brain initialization. Earlier versions checked all possible connections during construction of the brain and then checked the probability (specified in the input file) of that connection being created. The time complexity for this approach was $\boldsymbol{O}(n^2)$, where n represented the total number of cells in the network (or networks) that was being connected. The new version looked at desired number of connections first. Based on the required number of connections, it choose available cells randomly and created a connection between them. This algorithm approached $\boldsymbol{O}(n)$ complexity during execution. The other major optimization was to reduce the Message Bus load to produce less overhead. This was done by reducing message size, enabling the message packets to reach their destination quickly [45]. Figure 2.7 displays the results of optimizations to NCS in the parallel environment. The inset gives the same plot but on a logarithmic scale.

Various other biological experiments were performed with this new version of NCS. One set of experiments looked at how information maybe encoded using spike-timing and membrane dynamics of biological neurons [49, 50]. These experiments made comparisons with artificial neural networks (ANNs) to show that biologically based NCS was more robust and flexible and could utilize multiple sensory modalities requiring pattern recognition, speech comprehension, and path planning as humans could. Decomposed audiovisual recordings were given as input to laboratory derived parameters

for synaptic and membrane dynamics. These results were later on used in development of CARL [45].

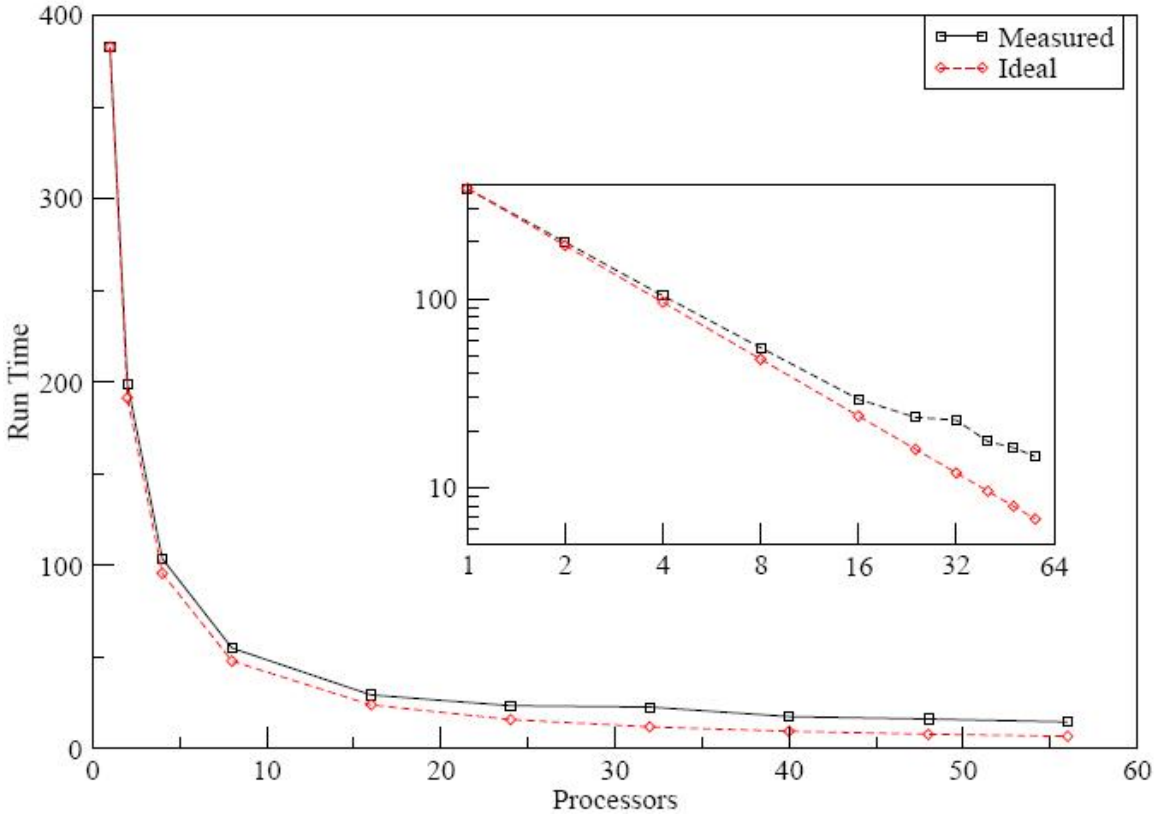


Figure 2.7 Latest optimizations to NCS achieve speedup closer to ideal. [32]

Another experiment [61] examined intra cortical and cortical-sub cortical "network of networks" involved in emotion, attention, sensory association, memory, reward, motor planning and theory of mind. This experiment tried to comprehend how these modules interact with each other, using NCS as a platform for experimentation. Using existing research parameters, models of 1,000 to 1,000,000 neurons were built which would reproduce and sustain auditory patterns with stability. A large inter-connected network resulted as the number of synapses was close to 600 million [45].

NCS Today

There have been some changes and bug fixes to NCS 5 since it was first released, but no major changes. As of 2005, NCS has all of the following functionalities implemented [24].

- **Compartments:** sampling frequency and membrane compartmental realism sufficient to capture biological response properties, arbitrary voltage- and ion-sensitive channel behaviors, and multi-compartmental models distributed in 3-D (dendritic, somatic and axonal systems).
- **Synapses:** short-term depression and facilitation [52], augmentation [70] and Hebbian spike-timing dependent plasticity [54].
- **3-D Connectionism:** a layout to easily allocate neurons into sub-network groupings, layers, column, and sheets separated by real micron- or millimeter spacing, with realistic propagation distances and axonal conduction speeds.
- **Parallelism:** an inherently parallel, efficient method of passing messages of synaptic events among neurons.
- **Reporting:** an efficient way to collect, sample and analyze selected compartmental and neuronal behaviors.
- **Stimulation:** ability to (a) specify fixed, standard neurophysiological stimulation protocols, (b) port signals from an external device, (c) export neuronal responses and await subsequent replies from external systems (e.g., dynamic clamps, in vitro or in vivo preparations, robotic emulations). Normally a stimulus is specified in the form of spike or firing probabilities.

- **Freeze/resume system state:** the ability to stop a running simulation and hibernate all hardware and software parameters into a binary file, for unpacking and resuming in later experiments.
- **Command files:** simplicity in generating and modifying scripts.

There are still many discoveries to be made in both biological and computer based artificial intelligence fields. NCS can be enhanced with new useful discoveries in biology. Biologists can perform experiments which might only be possible in a simulation environment, using NCS. With the help of new discoveries, both fields can accelerate to their respective goals of understanding the nervous system and creating the first artificial "humanly intelligent" social being.

2.2.2 Enhancements and Applications

Enhancements to NCS version 5

There have been a few major enhancements to NCS. One of the enhancements developed by Rich Drewes as his thesis work is a Python [16] based environment (and library) for creating, editing, submitting and analyzing simulations – called Brainlab [29]. As the brain model becomes large and complex, it becomes increasingly difficult to use a simple text editor. More often than not, these types of models involve repetitive patterns of inter-connecting brain objects. So, if there is an error in the input file, it might be tedious to find it. There is a facility in NCS for distributing the brain model code over multiple files and then using "INCLUDE" command to link those files. But it still is not easy. This is what the Brainlab was created for.

James King was responsible for maintaining NCS 5 code in 2004-2005. His thesis – "Brain Communication Server: A Dynamic Data Transferal System for A Parallel Brain

Simulator" (BCS) is an interface between NCS 5 and external programs [45]. It allows programs to communicate and send stimulus (data) and control signals to NCS 5. BCS is based on the concept of Single Socket Server (SSS). It is suitable for large simulations and communicating with many client programs, as it uses just a single port for each of those clients. In SSS, the external client program uses just a single socket to communicate with NCS via BCS. Reader and writer threads can be created for receiving and passing reports and stimulus, respectively, to NCS. This has been used by many other researchers in their experiments including in this thesis work [34, 45]. Figure 2.8 shows the design of Single Socket Server. Clients are connecting through a single port and the data objects (A, B, C and D) are managed by the server.

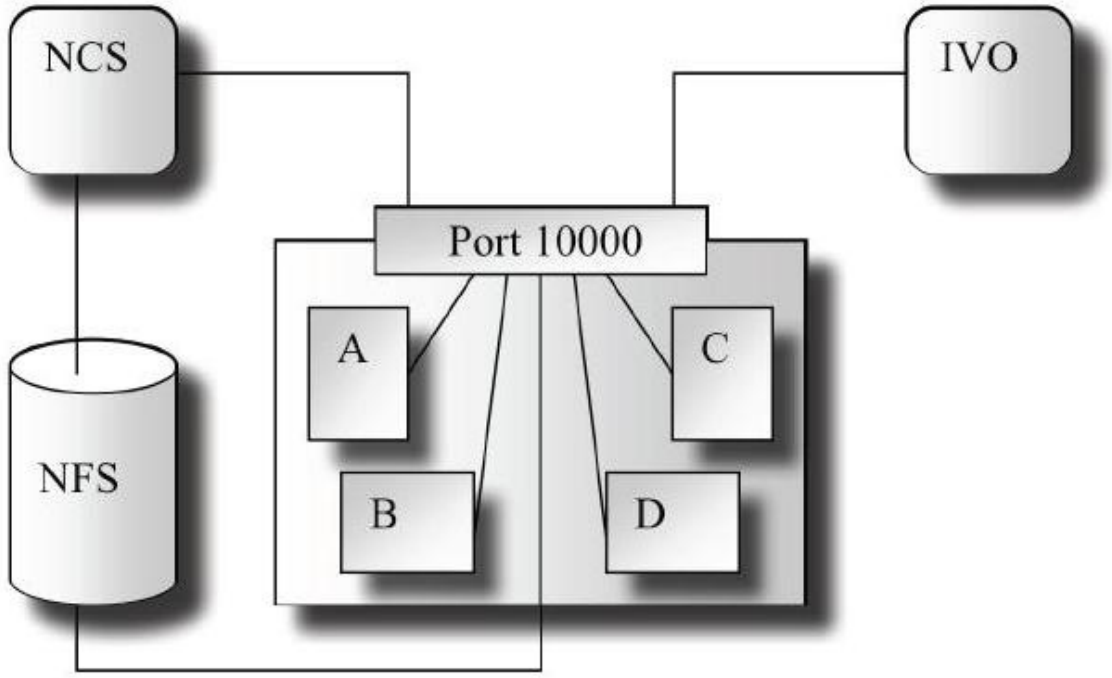


Figure 2.8 Architecture of BCS loop with Single Socket Server concept. [45]

James King also introduced numerous feature enhancements such as the efficient use of multi-compartment neuron models, synaptic augmentation property of cortex, changes

in the way NCS 5 creates some reports, ability to turn Hebbian learning on or off and an event section with the ability to load synaptic USE values from a file at a pre-determined point of time in a simulation [10].

Figure 2.9 describes the pipe-line model of Brainstem. Using the BCS (Brain

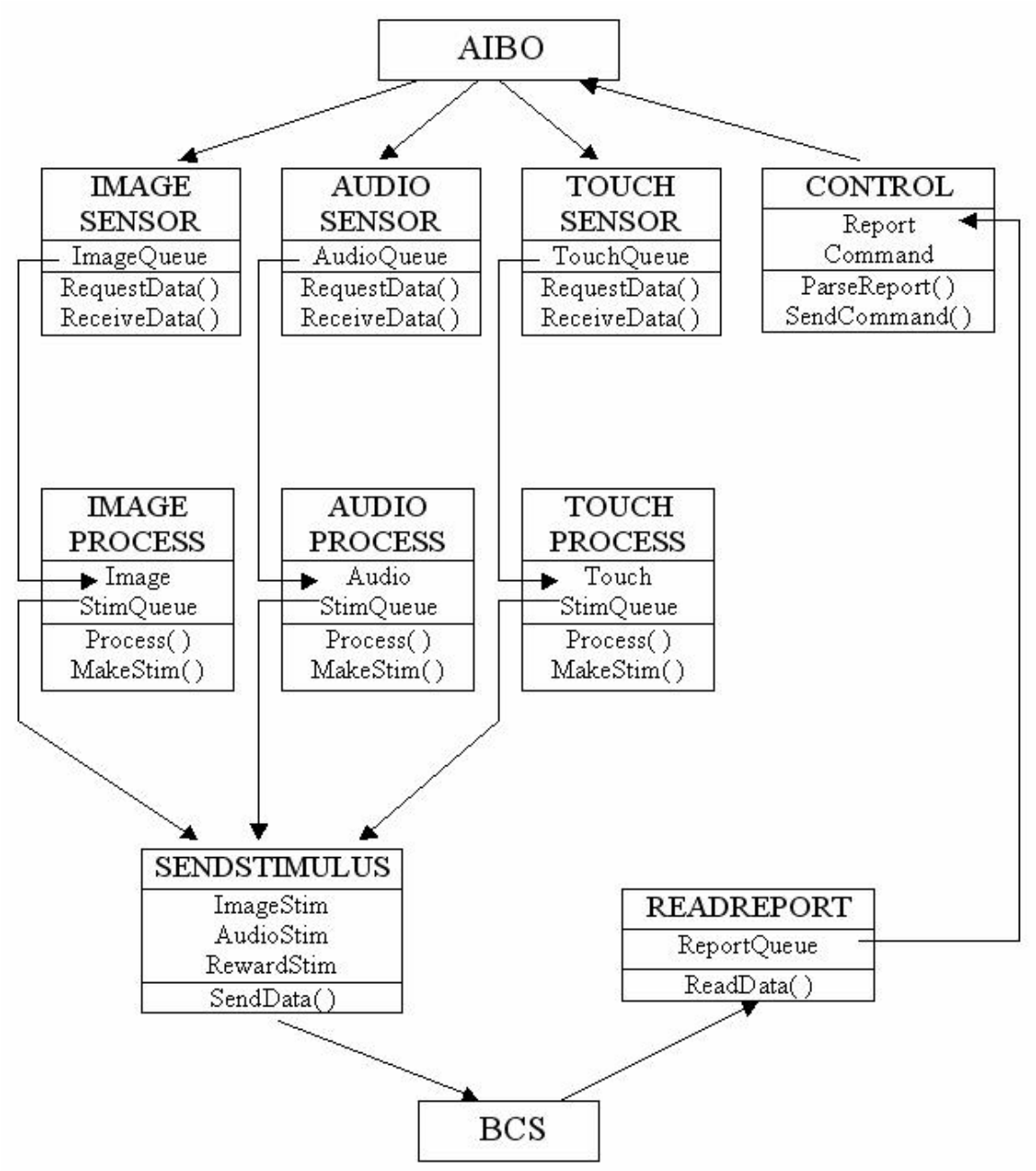


Figure 2.9 Pipe-Line model of Brainstem. [58]

Communication Server mentioned earlier, developed by James King for his thesis), Qunming Peng developed "Brainstem: A NeoCortical Simulator Interface for Robotic Studies" as his thesis work [58]. Brainstem is a program which co-ordinates communication between AIBO (an entertainment robot dog from SONY) and NCS 5 through BCS (also known as VoServer) [10]. This work was done in an effort to complete the loop between an autonomous physical agent (AIBO) and its thinking brain.

Applications

Numerous experiments and research projects have been performed using NCS. The majority of them are listed below:

- James Maciokas and other researchers used NCS to perform experiments on a hypothesis regarding entropy change in multimodal sensory integration and information transfer. This was done during 2002-2003 [48, 49].
- A web-based GUI environment was created by Waikul and colleagues in 2003 [69]. The environment had an editor for creating and modifying NCS input files and submitting those brain models for simulation.
- The first NCS-Robotics loop was completed and demonstrated by Juan Carlos Macera in 2003 [46, 47]. It consisted of a hierarchical robotic system (CARL) communicating with NCS for audio and video stimulus processing using Spectrogram and Gabor filters respectively.
- In 2004, Jake Blake and collaborators worked on speech perception which was simulated in a biologically realistic model of auditory neo-cortex [21].
- Brian Opitz in 2004 studied the effect of selective removal of channels from simulation on information transfer in a spiking neural model [57].

- Rich Drewes along with fellow researchers experimented using a genetic algorithm to evolve a network model for evolutionary autonomous agents for performing a delayed-matching memory task using a neural network with layer-structured visual cortex [29, 30].

2.2.3 Using NCS 5

At this point we are going to look at a basic method of using NCS in a Linux environment. We will begin with getting the source code, all the way to analyzing a simple report in MATLAB. Obviously there are a lot of ways in which NCS could be used. For example, Python based Brainlab, web-based environment, scripted pre-processor, automating input file generation and batch submission (by having a template input file and replacing tokens to create networks with slightly different parameters), etc.

The first step is to get latest working NCS source code. We use Subversion (SVN) for maintaining the source code and documentation. Source code may be obtained in two ways. A person can either get an account on the Cortex cluster with consent of responsible people in the BCL or get the source code straight off of the internet here [12]. In case of the former method, after getting an account setup on Cortex we use SVN to get the source code as follows –

svn co file:///home/svn/repository-new/ncs5

Doing this we obtain a directory named ncs5 with source code for the simulator and documentation.

The next step is to compile the source code, which may be tricky. We need the compatible version of MPICH. Currently we are using MPICH version 1.2.7p1. The Makefile situated in ncs5 directory has a host of options (switches and flags) for various

facilities. For a normal simulation those options would remain as they are. Only in case of special requirements would one change them. We need to check that the path to MPICH directory in the Makefile is correct. After everything is setup, we are ready to compile NCS version 5. Simply type in *make* and that should create an NCS 5 binary.

The next step is to create a brain model in an .in (input) file. If a job scheduling software is available for your cluster you can proceed to use it as you would for any other program binary. NCS can either be executed using MPICH (to run on multiple CPUs) or as a normal executable to run on the local machine. A sample command would look like this –

/opt/mpich-1.2.7p1/bin/mpirun –nolocal –np 16 –machinefile mach16.mpi ncs5/ncs5pe sample.in

where, *mpirun* is the MPICH program, *–nolocal* is a switch for MPICH and indicates not to run the simulation on the current node (or machine, which might head node of the cluster normally), *–np 16* means that the number of processor to use for the simulation are 16, *–machinefile mach16.mpi* gives the name of the file which has a list of names of the machines to use (this should have a list of machines consisting at least 16 CPUs in total), *ncs5/ncs5pe* is the name of NCS 5 executable along with its location, *sample.in* is the neural network input file.

Now the simulation is running and depending upon the input file we will get result in form of reports. A common report is the voltage report for a cell group consisting of either 1 or multiple cells. The report file has membrane voltages present in it for each simulation time tick (i.e. for each step of thinking brain). In the above scenario, these report files are created on the cluster and need to be copied to a local machine for

analyzing in MATLAB. Neuroplot [14] is an excellent tool for such analysis purpose. This completes a simple guide for using NCS to simulate neural networks.

2.2.4 Structure of NCS 5 Input File

The structure of the NCS 5 input file is simple. It is based on a hierarchical structure of sections (or objects). NCS has objects representing biological structures, like a neuron, synapse, compartment, etc., which are used to mimic their behavior using the right combination of parameter values. We will not be going into details of the file structure, but just look at a couple of sub-sections and get a feel for the structure. The extension of input file is *.in* and hence is also called as an .in file in general usage. We will be observing that convention from here on.

There are keywords in NCS and generally all of them appear in upper case. The .in file consists of a number of sub-sections. The sub-section starts with a line containing its name in all upper case letters. The end of a sub-section is denoted by a line having the same name that appears at its start, but with a prefixed string – **END_**. The first sub-section in an .in file is **BRAIN** section. It is the top-most section in the hierarchy of the brain structure. Global parameters and other basic sub-sections are referenced in this section. A sample **BRAIN** section is show in Figure 2.10.

The **TYPE** name gives a name for the network model and **JOB** describes a name for the execution run. **SEED** is used by the Random Manager and is used by other sub-sections too (wherever applicable) if a local seed is not present. References to other sub-sections can also be seen. **COLUMN_TYPE**, **STIMULUS_INJECT** and **REPORT** are sub-sections defined after the **BRAIN** section definition. The sections essentially contain key-value pairs. Keywords of NCS are the keys and values or section (object) names are

there values. **CONNECT** keyword is used to connect two cell groups in the network using a **SYNAPSE** object (here **synEE_V2**).

```
BRAIN
 TYPE                  AIBO
 JOB                   proto7
 FSV                   1000
 DURATION              1.5
 DISTANCE              NO
 SEED                  -51
# ============== this is a comment ================
 COLUMN_TYPE           AIBOStim_1
 COLUMN_TYPE           AIBOStim_2
 STIMULUS_INJECT       AIBOStim_inj_E1_AIBO_1
 STIMULUS_INJECT       AIBOStim_inj_E1_AIBO_2
 REPORT                L1_E1_Voltage
 REPORT                L2_E1_Voltage
 CONNECT
                       AIBOStim_1  AIBOStim_layer_1  E1    somaE
                       AIBOStim_2  AIBOStim_layer_2  E1    somaE
                       synEE_V2    0.10    0
 CONNECT
                       AIBOStim_2  AIBOStim_layer_2  E1    somaE
                       AIBOStim_1  AIBOStim_layer_1  E1    somaE
                       synEE_V2    0.10    0
END_BRAIN
```

Figure 2.10 The **BRAIN** section of a sample NCS input (.in) file.

Now we will discuss the basic hierarchy found in the .in file. **BRAIN** is the top-level section in an .in file. A **BRAIN** section consists of **COLUMN** objects which in turn consist of **LAYER** objects. **LAYER** further has **CELL** objects, which themselves consist of **COMPARTMENT** objects. In the hierarchy mentioned above, there can only be one **BRAIN** section, but other sections or objects can be one or multiple. A **CONNECT** statement can be used for connecting cell groups visible from a given sub-section. For example, to connect cell groups in the same layer, we can have **CONNECT** defined in that **LAYER** section definition. But if the cell groups to connect are located in different

layers present in different columns, then the **CONNECT** statement needs to be defined in the **BRAIN** section, as seen in Figure 2.10.

Normally object names are defined in the **TYPE** keyword of their respective section definition. For example: **AIBOStim_1**, **L1_E1_Voltage**, in Figure 2.10. Another thing to note is that NCS interprets everything after # character, until an end of line is detected, as a comment. A comment is a line used by modelers for their documentation purposes and is not processed by NCS simulator.

We can define different sections in different .in files and use the keyword **INCLUDE** along with the .in file name. A thing to remember is that the .in file with **BRAIN** section is the main file. So while simulating a network, only the .in file containing the **BRAIN** section should be given as a parameter. Other .in files should either directly or indirectly be included in that main file.

This concludes the section which briefly explains the structure of an NCS brain model input file. For further information, NCS documentation is located on the BCL webpage [10] along with a complete sample .in file.

2.3 Modeling Intelligence

Modeling intelligence successfully has proven to be a bigger challenge than anticipated earlier. That is why we still see practical social robots only in movies like Artificial Intelligence: AI and I, Robot. There have been various approaches towards this problem. Earlier efforts resulted in the field of Artificial Intelligence. Due to the

shortcomings in AI [25], a shift in thinking resulted in Artificial Neural Networks and Biological Neural Networks.

2.3.1 Artificial Intelligence

As discussed earlier, the term Artificial Intelligence (AI) was created in 1956 [25, 62]. AI was the next step in automation of various specialized and daily tasks. AI has applications in fields of Pattern recognition, Computer vision, Virtual reality, Diagnosis of various kinds, Game theory, Strategic planning, Natural language processing, Non-linear control, Robotics, and many more.

The earliest AI systems were the Expert systems. Inference engines with forward and backward reasoning were the basis for these systems. Classifier based systems are another variety of AI systems involving supervised learning. Classifiers are basically functions that can be tuned for certain problems according to some examples (which in fact are patterns of some form). Conventional AI involves methods from Machine Learning [55] involving formalism, search, planning, optimization, logical inference, data mining, and statistical analysis to name a few. It consists of mainly following methods:

- Expert Systems which reason to reach a conclusion. They can analyze large amounts of information and provide a conclusion for the same [75].
- Case based reasoning organizes a set of problems and answers in form of cases. When a problem is presented to a case based system, it searches through its database of cases and picks an exact or closely related case and provides its solution with some modification if needed [76].

- Bayesian network (belief network) use a probabilistic graphical model representing a set of variables and their respective probabilistic independencies [77, 78].
- Behavior Based Artificial Intelligence is a technique for building AI systems modularly by decomposing of intelligence into modules [79, 80].

The major problem with AI systems is that they are not generalized enough to handle real world scenarios which are fuzzy and full of noise. More often than not, they are deigned to work in a controlled environment which is far from real world interactions. Due to these shortcomings there have been on and off views of the field and AI winters, meaning general pessimism resulting in reduced funding research in AI and termination of serious research [25]. There have been two instances of such scenarios, first from about 1974 – 1980 and second from 1987 – 1993. Never the less, field of AI continues to move on with funded projects happening every now and then. There is still optimism about the field.

2.3.2 Artificial Neural Networks (ANNs)

An Artificial Neural Network (ANN, also known as Simulated Neural Network) is mathematical computational model based on biological neural networks [20]. It consists of an inter-connected group of artificial neurons which essentially sums its inputs and applies a transfer function (normally a sigmoid) on it. These model actual biological neurons at a very high level of abstraction and hence are termed as Artificial Neural Networks. Biological Neural Networks are made up of biologically realistic neurons [81]. An ANN traditionally finds applications in regression analysis, classification, data

processing and mining, etc. which are very broad fields. They are now often used as controllers too. Figure 2.11 shows a simple example of a 3 layer feed-forward ANN.

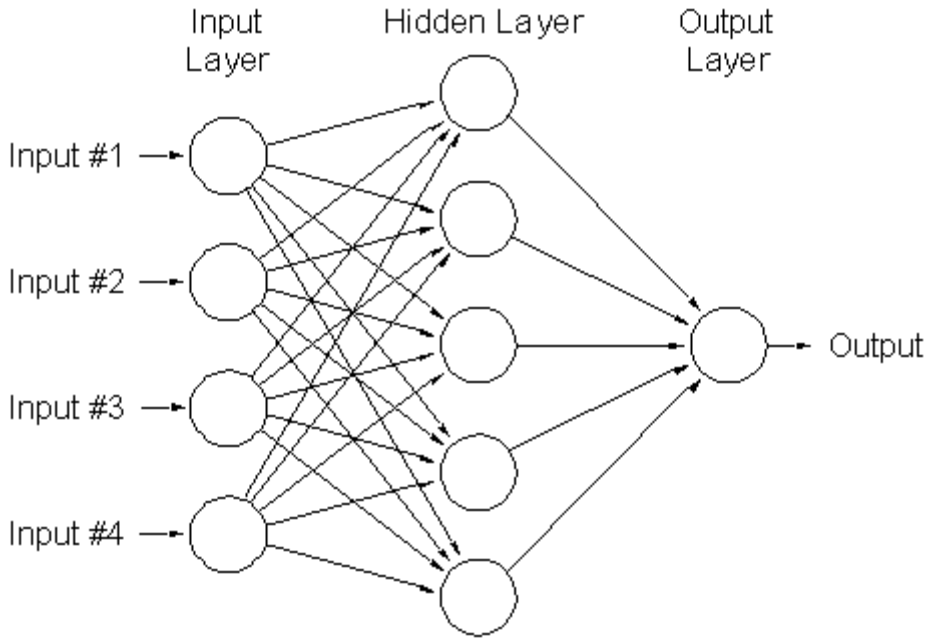


Figure 2.11 A representation of a simple 3-layer feed-forward ANN with 4 inputs, 5 hidden nodes, and 1 output. [1]

An ANN simply defines a mathematical function. Learning happens in form of supervised, unsupervised, and reinforcement learning. There are various types of Neural Networks. Some of them are Feed-forward neural network (FNN), Radial basis function network (RBFN), Hopfield network, Recurrent Neural Network (RNN), and Associative neural network (ASNN). Each one has its area of application. RBFNN are universal function approximators [83, 84]. RNN allow internal state and have been used as universal Turing machines [85]. Most of these networks do not provide sufficient level of complexity and generalization for a socially interactive intelligent agent.

Spiking Neural Networks are more closely based on biology. Besides having neuronal and synaptic states, they incorporate the concept of time into their operating model. Neurons in these models only fire when the membrane potential of neuron reaches the firing threshold. When this state is reached, the neuron fires and sends a spike to all the neurons it connects to, via synapses. This spike, depending upon synapse and other parameters, may induce a spike in the following connected neuron. Information may be encoded in form of frequency of firing or timing of actual spikes which makes Spiking Neural Networks attractive to use as basis for further experimentation.

Role of NCS

NCS has been based on Spiking type of neural networks. NCS allows for biologically realistic spiking neural networks. Details about NCS have already been discussed in Section 2.2. Using NCS we attempt to develop a principle which is based on the human neural system. Although the task at hand seems impossible as we know so little about human biology, with help of artificial simulation and ongoing biological explorations, we will achieve that principle in the near future. This thesis addresses a plausible first step in this principle and also looks at the performance of NCS models for real time response.

Chapter 3

RAIN – Recurrent Asynchronous Irregular Network

3.1 Overview

There is a presence of ongoing activity in a human brain as long as the human is alive. This activity can be quantified by Electroencephalography (EEG) [8], which is a tracing of electrical activity occurring in the brain, recorded by placing electrodes on a human scalp. This electrical activity has been classified into different types of waves. All these waves are mostly oscillating between 1-20 Hz when recorded over a scalp [23].

Figure 3.1 shows a sample of alpha (7 to 12 Hz, associated with a relaxed awake state) and beta (13 to about 20 Hz associated with an active concentration state) waves [15].

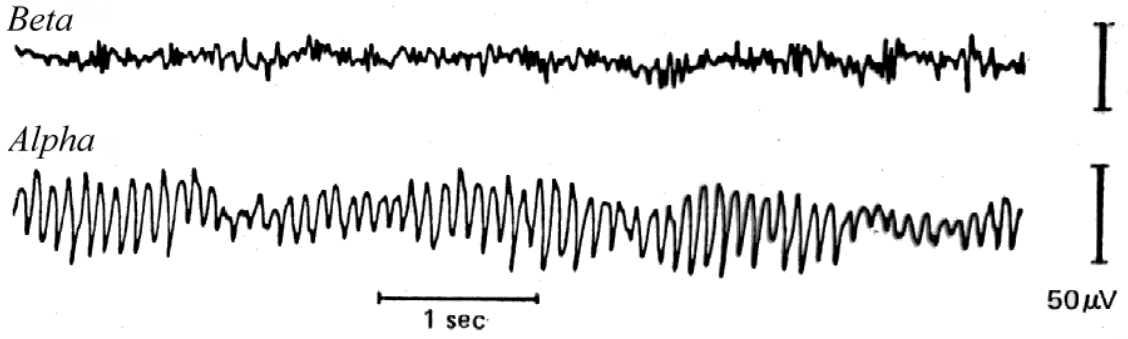


Figure 3.1 Alpha and Beta waves. [5]

RAIN is an acronym for Recurrent Asynchronous Irregular Network first coined by Philip Goodman at the Brain Computation Lab. In simple terms, RAIN represents a

spiking activity exhibited by an NCS neural network, firing in a pattern which is recurrent, asynchronous and irregular. This activity is similar to the random "noise" activity seen in an EEG of human brain. Hence this is an attractive paradigm for further experimentation of modeling human intelligence and memory. Figure 3.2 shows an image depicting RAIN activity (about 16 Hz) obtained using NCS 5 with a 4000 integrate and fire (IF) neuron network model.

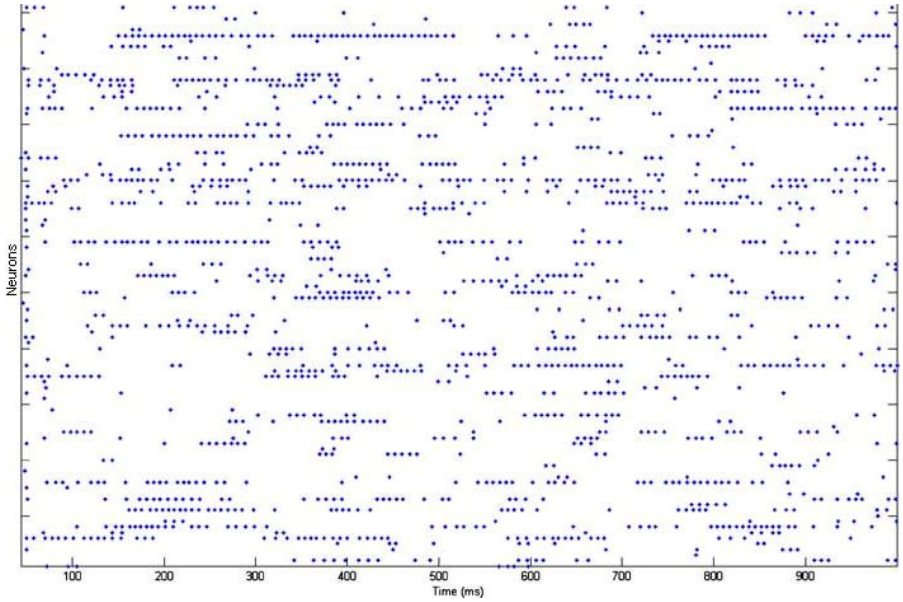


Figure 3.2 A typical RAIN activity. [24]

3.2 Initial Neural Network Model

Several of us were working on the concept of modeling Neocortical Lock and Keys to implement an information encoding scheme resulting in a memory model which can learn using spike timing dependent plasticity (STDP) [27]. During this time (late March 2006), we were invited to be a part of a review paper about biologically realistic simulators [24].

To have credibility, the invited simulators needed to demonstrate certain abilities enlisted by benchmark models based on a paper by T. P. Vogels and L. F. Abbott [68].

The benchmark model we chose for NCS 5 was for neural network models based on integrate and fire (IF) neurons with conductance based (COBA) synapses (COBA model of T. P. Vogels and L. F. Abbott [68]). The model consisted of a network of excitatory and inhibitory neurons, connected via conductance-based "exponential" synapses (instantaneous rise, exponential decay). For sufficiently large size network (more than 4000 neurons), this model displays self-sustained irregular states.

Details and a brief description of the parameters are as follows [68]:

- Total number of neurons in the network is 4000 with a 4:1, excitatory to inhibitory ratio.
- Area of cell (neuron) is 20000 μm^2, ignored in our case since we chose not to have distance in the model.
- Neuron membrane time constant equals 20 ms. In NCS 5; this parameter (**TAU_MEMBRANE**) along with **R_MEMBRANE** is used to calculate membrane capacitance [82].
- Specific capacitance of 1 $\mu F/cm^2$ was achieved by adjusting **R_MEMBRANE** value [82].
- Leak conductance should be $5e^{-5}$ S/cm^2. It is used to calculate the leak current when the compartment updates its membrane voltage in absence of a voltage clamp. This is used in conjunction with leak reversal.
- Leak reversal is supposed to be -60 mV.
- Synapse parameters:

  - Excitatory conductance = 6 nS.
  - Inhibitory conductance = 67 nS.
  - Excitatory reversal potential = 0 mV.
  - Inhibitory reversal potential = -80 mV.
  - Excitatory time constant = 5 ms.
  - Inhibitory time constant = 10 ms.
- Connectivity parameters: random connectivity, with probability of connection as 2%.
- A random stimulation during initial 50 ms is needed to set the network in active state.
- IF model used:
  - Firing threshold is -50 mV. When the compartment's membrane voltage reaches threshold, it begins a spike shape (i.e. the neuron fires).
  - Reset voltage is -60 mV. In NCS 5, this is represented as **VMREST** in **COMPARTMENT** section [82]. It represents the starting voltage of the compartment. When the compartment is at rest, it is at this voltage level.
  - Compartment refractory period is 5 ms and it is clamped at reset voltage.

Building the Model for NCS 5

Apart from looking at benchmark model specification, we also looked at the source of benchmark models, the paper by T. P. Vogels and L. F. Abbott [68]. The following were the major details of parameters in our model and the modified parameters from above initial specification:

- As per the paper [68], we created three cell groups in the network, E1, E2 and I. E1 had 33 excitatory cells, E2 has 7967 excitatory cells and cell group I has 2000 inhibitory cells. Thus the network had 10,000 neurons in total.
- All the cell group combinations were connected (viz. E1-E1, E2-E2, E1-E2, E2-E1, E1-I, I-E1, E2-I and I-E2), except for self connections in I.
- As mentioned earlier since we chose not to include distance in our model, area of cell has been ignored.
- With respect to synapse parameters: excitatory time constant and inhibitory time constant, NCS 5 has a facility for specifying that in form of values in a text file. This file is defined in section of synaptic waveform known as **SYN_PSG** [82]. The time constants of 5 ms and 10 ms for **FSV** of 10,000 have been plotted in Figure 3.3 as EPSG and IPSG, respectively [82].
- **FILE_BASED** type of stimulus in NCS 5 was used. This required a file with values (1.0, meaning 100% probability of firing) representing stimulus. A pure random Poisson type of stimulus was used [82].
- Compartment refractory period is implemented by the **SPIKESHAPE** section [82]. In NCS, there is a concept of Frequency of Sampling Value (**FSV**). It is used in scaling the duration of simulation (in terms of time ticks, or the time steps of Do Think loop of brain). E.g.: If duration in .in file is set as 2.4 seconds and FSV is 1,000, then the total time ticks would be multiplication of duration and FSV which is 2,400 time ticks. So in our case, FSV is set at 10,000 and refractory period is 5 ms or 0.005 seconds. So the number of values in the spike shape waveform are 50 (0.005 * 10,000 = 50). NCS ensures that when a spike occurs,

this spike shape values are used to represent the compartment voltage for 50 ticks. So for this period of 50 ticks, only one spike is present and hence the refractory period for the compartment is enforced.

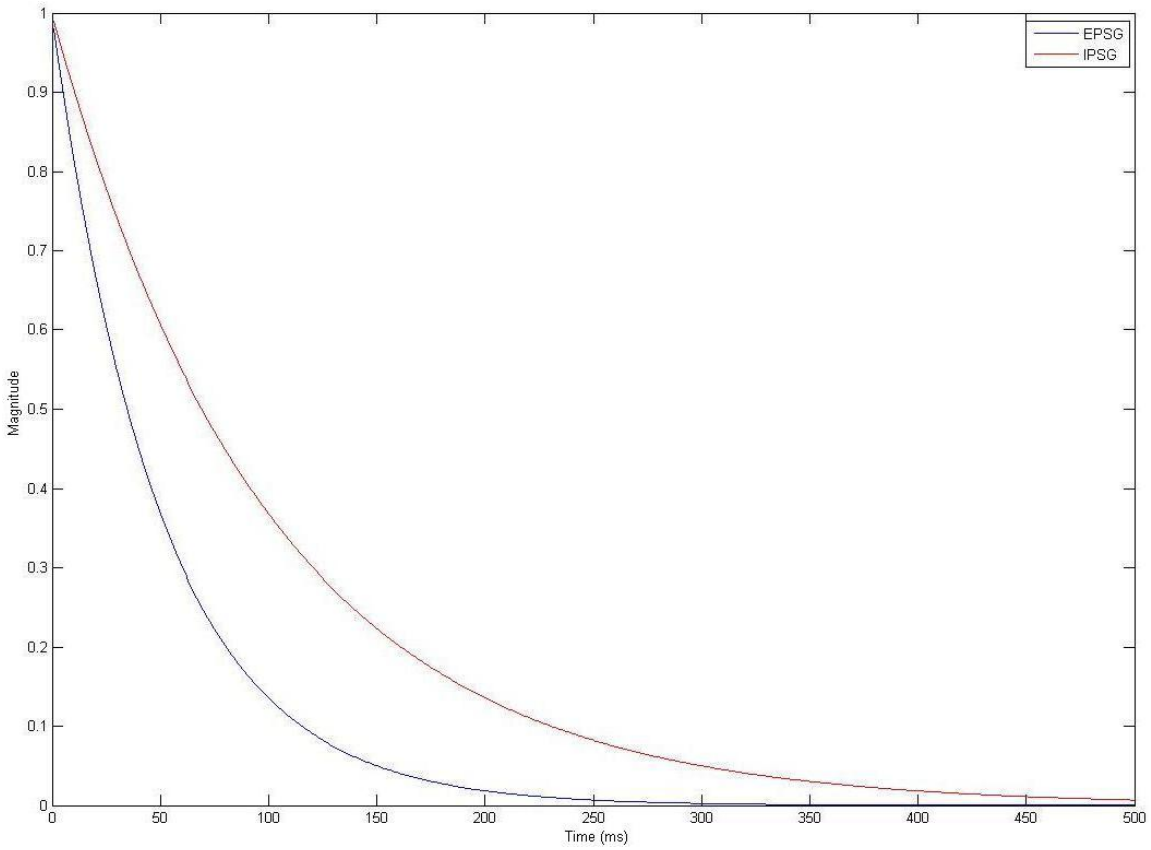


Figure 3.3 EPSG and IPSG synaptic waveforms with 5 ms and 10 ms time constants, respectively.

Figure 3.4 represents the 10,000 neuron architecture. This is based on the initial configuration parameters from T. P. Vogels and L. F. Abbott paper [68].

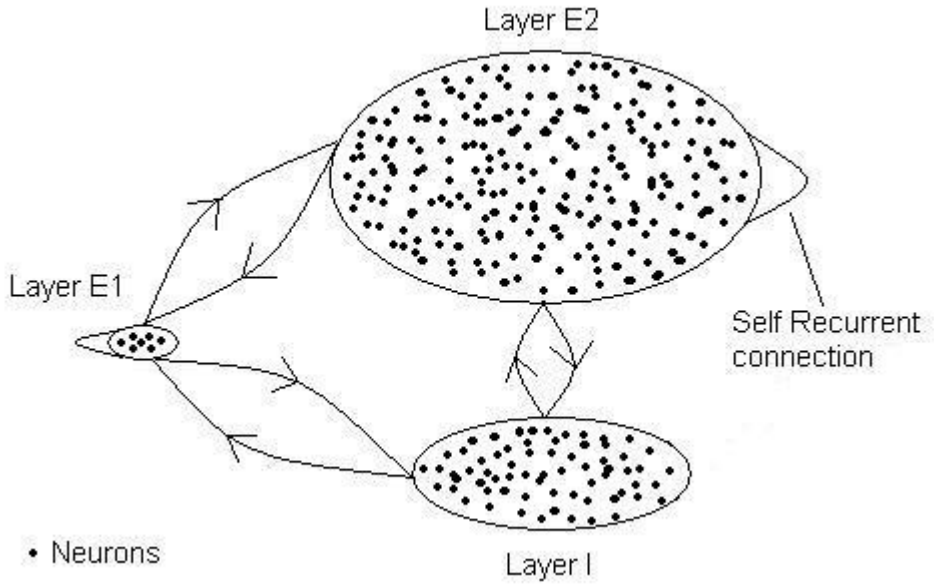


Figure 3.4 Architecture of the neural network built in NCS 5.

3.3 Recalibration of Initial Neural Network Model

Various parameters were tuned in the process of generating RAIN activity. The ones worth mentioning are synaptic conductance (both excitatory and inhibitory), synaptic delay, synapse absolute use and connection probability. Also along the way, few changes were made to NCS 5 code to meet desired setup for this particular project. The first such change was commenting out a switch (or a flag to the compiler) statement in the NCS 5 make file, *CFLAGS += -DSAME_PSC*. This flag forced the simulator to use same synaptic waveform file for both excitatory and inhibitory synapses, causing both synapse types to have the same time constant of 5 ms. This was discovered during an initial experiment.

3.3.1 Coding

Synaptic conductance was the most changed parameter. There was a need to explore ranges of combinations of excitatory and inhibitory synaptic conductance. This type of experiment only had change of synaptic conductance values across a number of .in files. Python scripts were written and used to generate the large number of input files which were identical except for the synaptic conductance. A template .in file was read in by the script and performed parameter replacement. Python scripts were also used in orchestrating the automated execution of experiments.

Besides Python, quite a few MATLAB scripts were written to perform analysis on voltage, fire count, connection map and synaptic conductance types of reports. There were a few code changes to NCS 5 also (like the correction of floating point approximation in synaptic delay parameter).

3.3.2 Recalibration and Initial Results

As mentioned earlier, synaptic conductance was the parameter that was changed most often. With these parameters we performed some simulations. By varying only synaptic conductance (**MAX_CONDUCT** value in **SYNAPSE** section) for excitatory and inhibitory synapses, we either had the network not firing (due to strong inhibition) or just going epileptic (a brain disorder involving recurrent seizures) due to strong excitation. This is shown in Figures 3.5 and 3.6. These figures have been obtained using the Neuroplot MATLAB script [14].

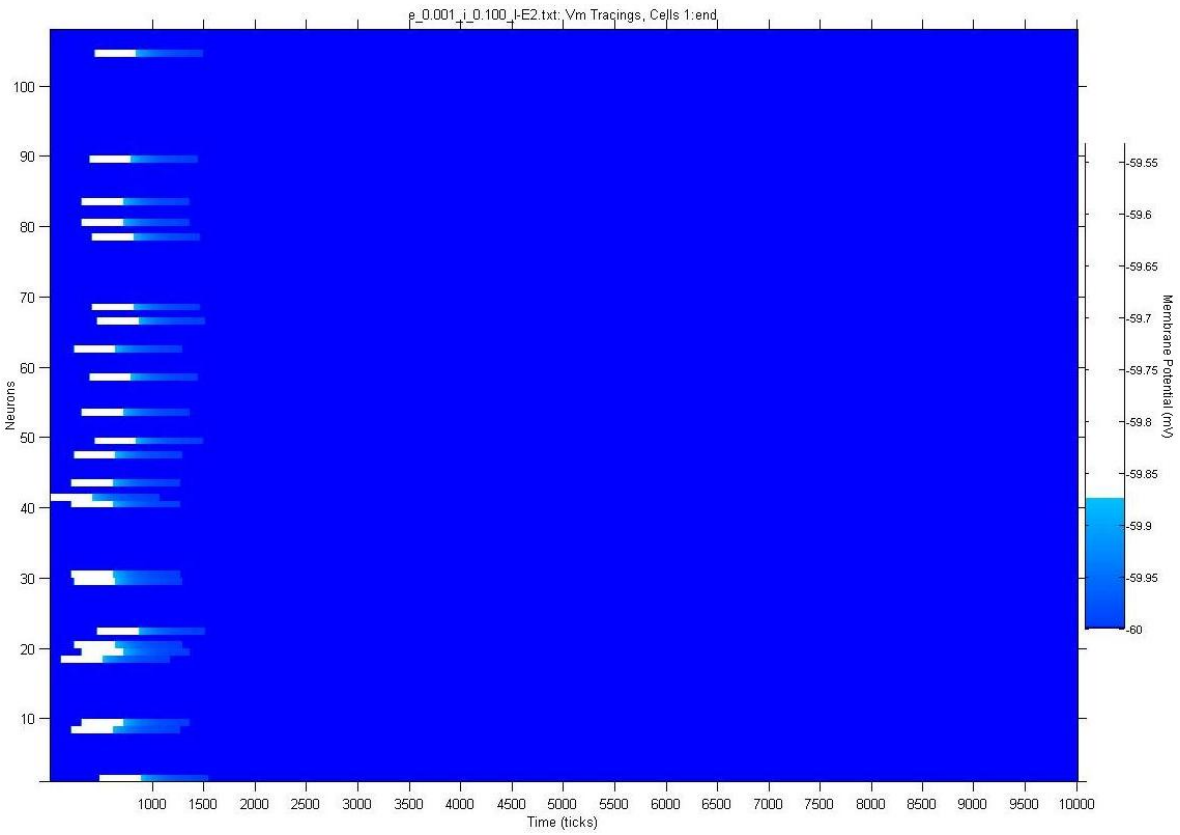


Figure 3.5 Few cells from E2 cell group indicating network has failed to produce any spikes.

The plots give voltage values for reported neurons (y-axis) over the reported duration (x-axis in terms of time ticks). The color bar near the right border of the image gives voltage value in mV.

The synaptic delay parameter had a significant effect. Changing this value affected the state of the network from epileptic to strongly inhibited and other similar transitions. Later on this value was settled at 0.0000000001 which was interpreted as almost zero.

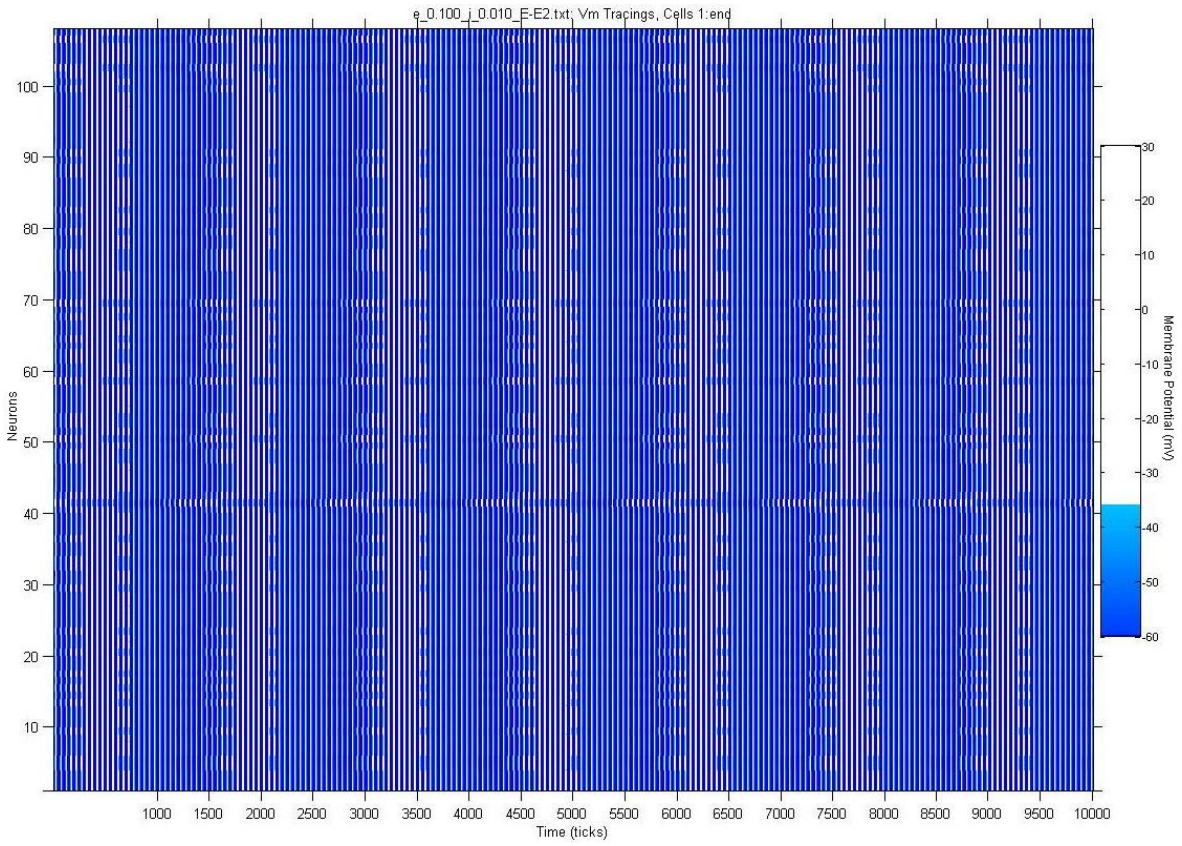


Figure 3.6 Few cells from E2 cell group representing continuous firing (spiking) state of the network.

We then decided to set up a new model. The total number of neurons in the network was reduced to 4000 as per initial model specification. Ratio of 4:1 between excitatory and inhibitory neurons was maintained. Correspondingly the number of neurons in excitatory cell group E1 was 14, excitatory cell group E2 was 3186 and inhibitory cell group I was 800. Also, the input stimulus file was of random Poisson type and changed to consist of 14 columns, one for each of the 14 cells in the E1 cell group. The file had 50 rows indicating input for 50 time ticks. The file had a single value of 1.0 (1.0 representing 100% firing probability), occurring once but randomly during the 50 time ticks for each of the 14 E1 cells. Figure 3.7 represents the architecture of the neural

network that was now used in experiments. From the figure, we also note that I-I connections were now used.

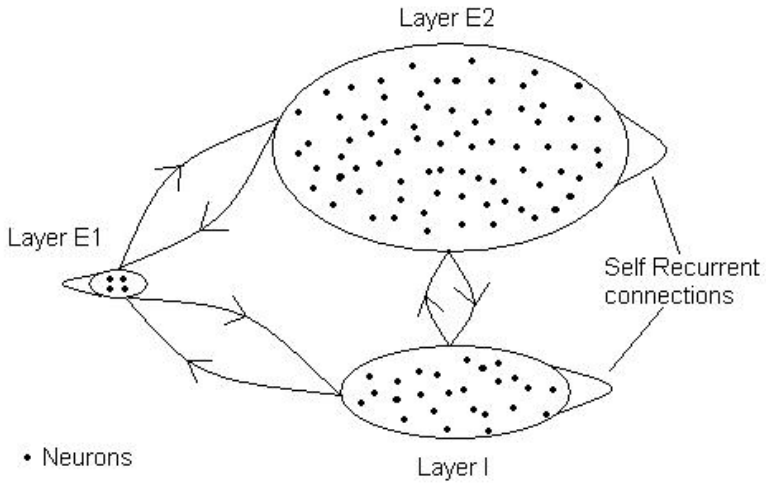


Figure 3.7 Recalibrated architecture of the neural network built in NCS 5.

Figure 3.8 gives the voltage state of cell group E2 after one of the experiments.

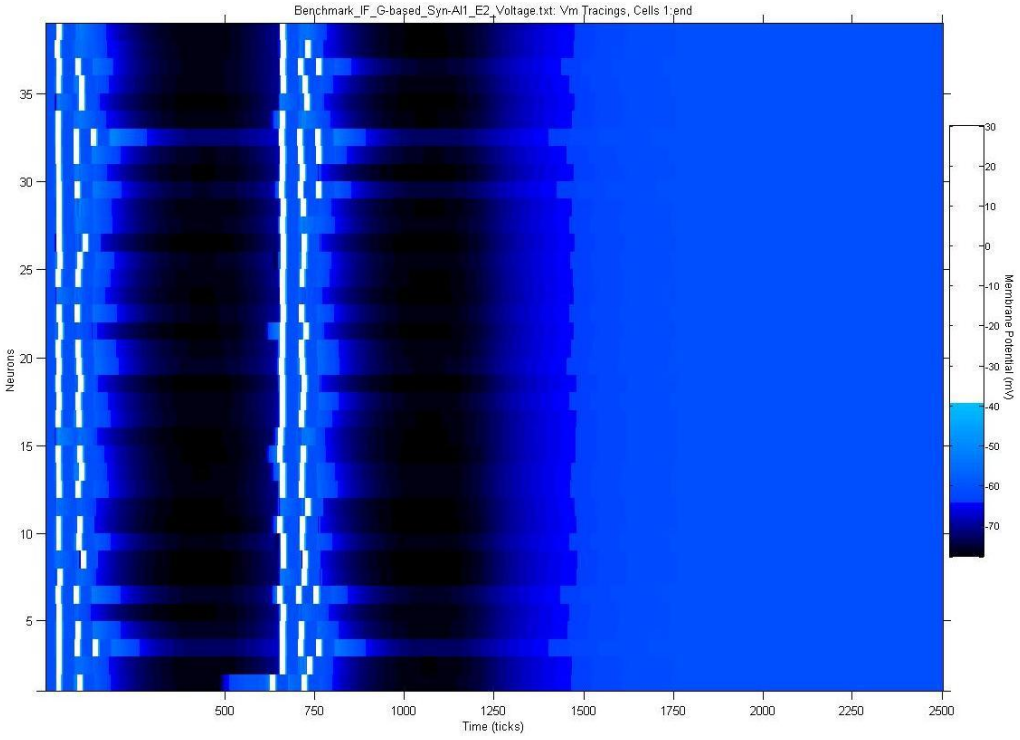


Figure 3.8 A voltage report of few cells from E2 cell group with initial burst of spikes.

As seen in the figure 3.8, there were initial signs that E2 (and the whole network correspondingly) was trying to fire but was stopped by strong inhibition (black region) coming in from cell group I. Thus we have a couple of waves of short spike bursts indicated by white rectangular blocks, and then the network dying out, certainly not what we wanted.

The next thing to look at was how NCS was setting up connections between cell groups. Default connection distributions followed an equal distribution algorithm. That means, all cells in the destination cell group were guaranteed to get almost equal number of connections from source cell group cells on average. It was suggested that we use a binomial distribution connection algorithm.

Figure 3.9 depicts the histogram of connections from cell group E2 (3186 neurons) to

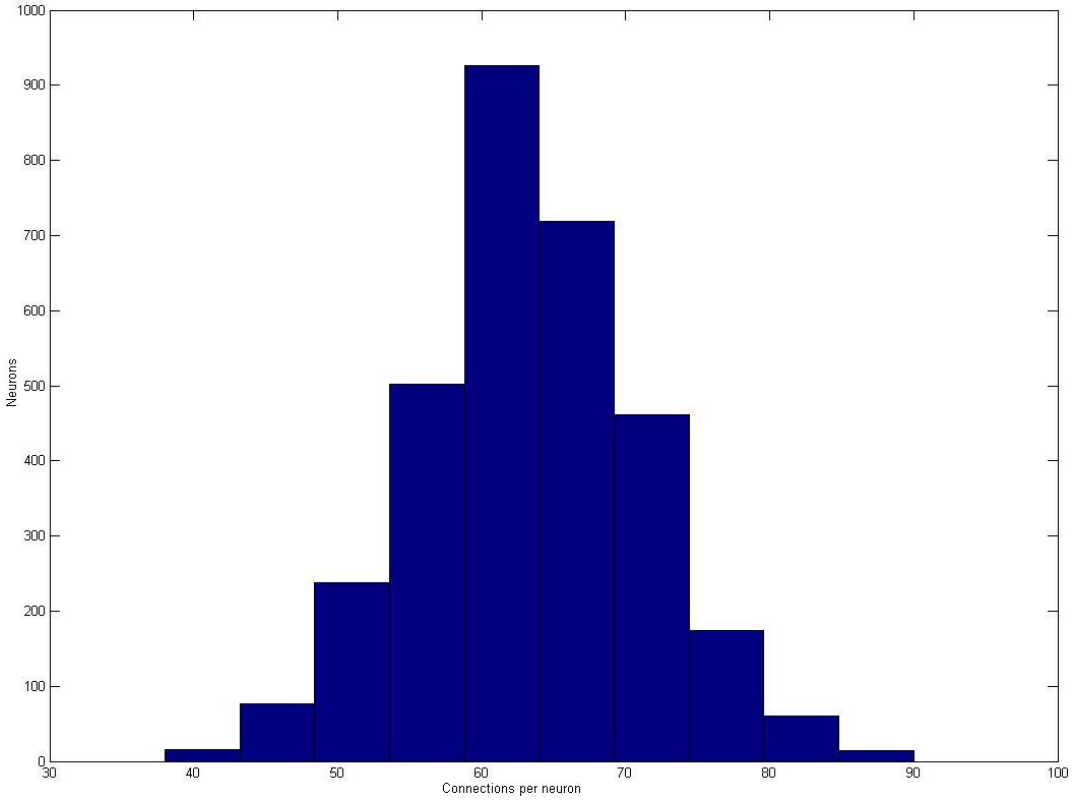


Figure 3.9 Histogram of binomial distribution algorithm for E2 to E2 self connections.

itself (self connections). Y-axis has number of connections made and x-axis has number of neurons from destination cell group. From the figure it can be seen that the number of connections made for neurons in the destination cell group (E2 in this case) are following a binomial distribution.

But the binomial distribution algorithm approach did not produce the expected result either. The model was checked and cross-checked against expected configuration results. For example, total number of synapses was one such parameter to be cross-checked. Various ratios of excitatory and inhibitory synaptic conductance were tried. Performing several experiments lead to a behavior similar to RAIN but not exactly RAIN, shown in Figures 3.10 and 3.11.

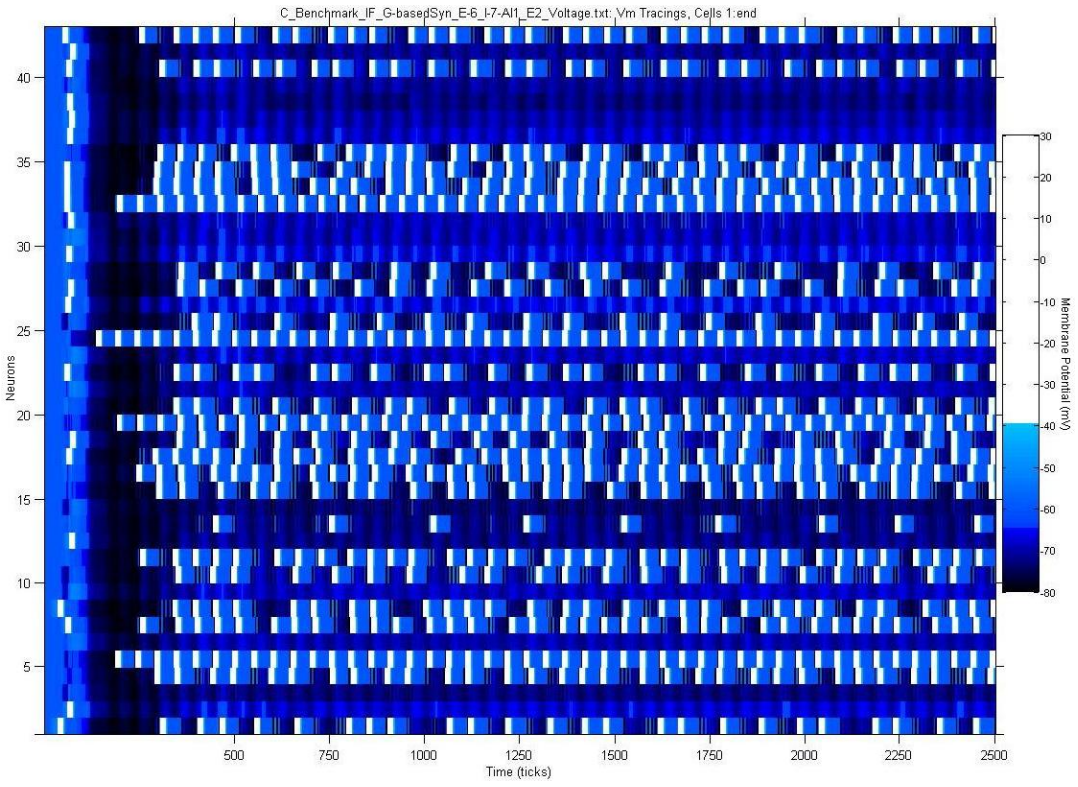


Figure 3.10 Typical voltage report plot of E2 (subset of all cells from cell group E2).

Figure 3.11 shows the membrane voltage tracings for the first 2 cells in cell group E2. The first cell spikes once initially and then quiets down due to initial inhibition where as second cell continues spiking at a fast rate. Here initial inhibitory wave is able to shut off a few cells but other cells continue to fire due to high excitatory conductance. Now we are seeing sustained activity but at higher repetitive firing rates. Excitatory and inhibitory synaptic conductance was 42 nS and 245 nS respectively.

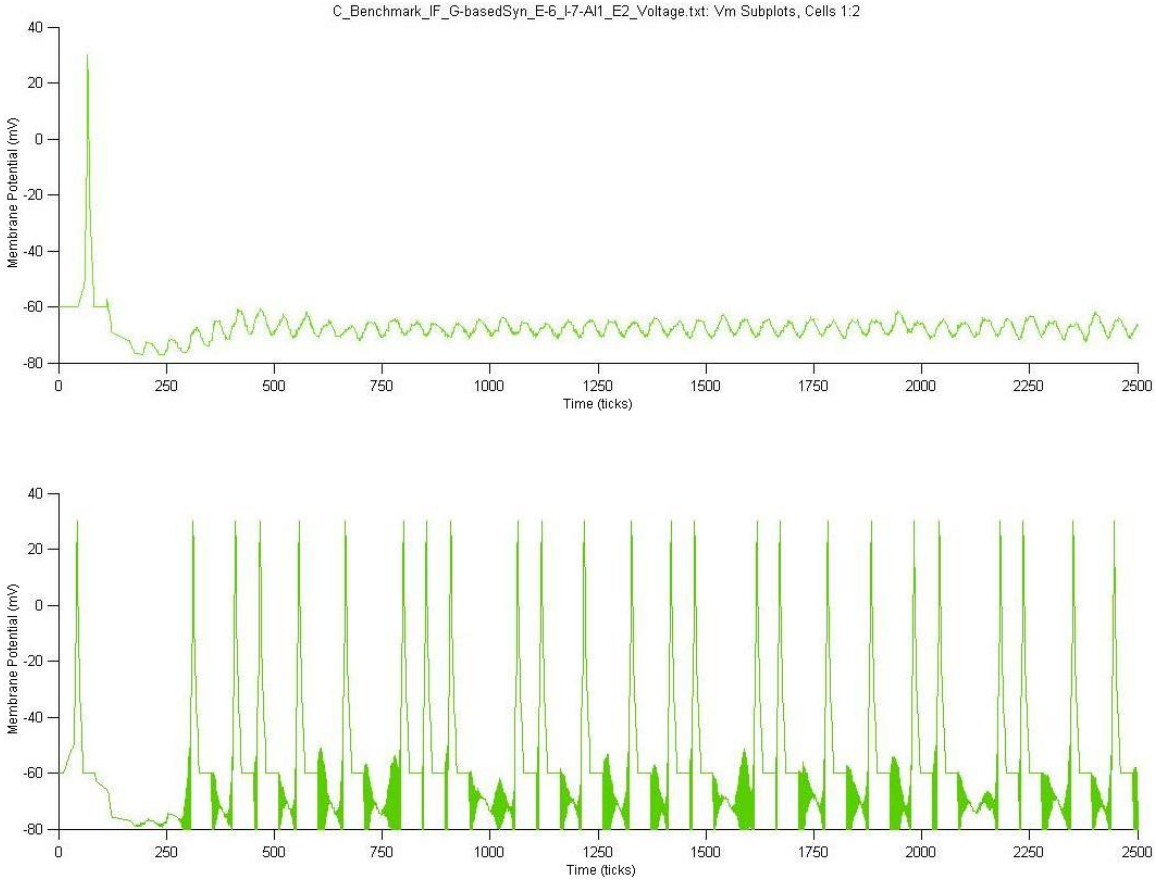


Figure 3.11 Membrane voltage tracings for first 2 cells in E2 obtained using Neuroplot.

3.3.3 Final Setup

We were now close to achieving RAIN behavior. Fine tuning of parameters (mostly synaptic conductance again) resulted in desired RAIN behavior. The final configuration

was having **FSV** at 10,000. Random Poisson stimulus with 14 cells as earlier was kept same. The stimulus current had amplitude of 3 nA and was given to cell group E1 during the initial 50 ms period of simulation. Distance was still off. Network was fully connected with nine connections at 2% probability.

Network size was 4000 (E1 = 14, E2 = 3186, I = 800). Compartment and synapse parameters were same as in the benchmark model specifications. Synaptic delay was almost zero seconds. Excitatory synaptic conductance was 10 nS and inhibitory synaptic conductance was 100 nS, a ratio of 10, matching the one given in benchmark model specification. Synaptic absolute value was 0.250. Short-term and long-term synaptic dynamics were not defined as per the benchmark model and paper [68]. We will discuss those final results in Chapter 5.

Chapter 4

Benchmarks for NCS 5

Benchmarks for NCS 5 are necessary to understand system performance and requirements of hardware for applications to operate in real time. At the time, there was a grant for purchasing new hardware. Since a fully functional cluster (Cortex) was already available the question was what kind of new hardware would be most beneficial. Recently a paper on Virtual Neuro-Robotics (VNR) was published in a new online journal [34] by researchers at the BCL (details about the system can be found here [18]). Since this involved a fully operational system (NCS 5 and a virtual robot), it was a good practical system to perform the benchmarks on. Using the results of this benchmarks as indicators, new hardware was to be purchased.

4.1 System Setup

4.1.1 Hardware and Software Involved

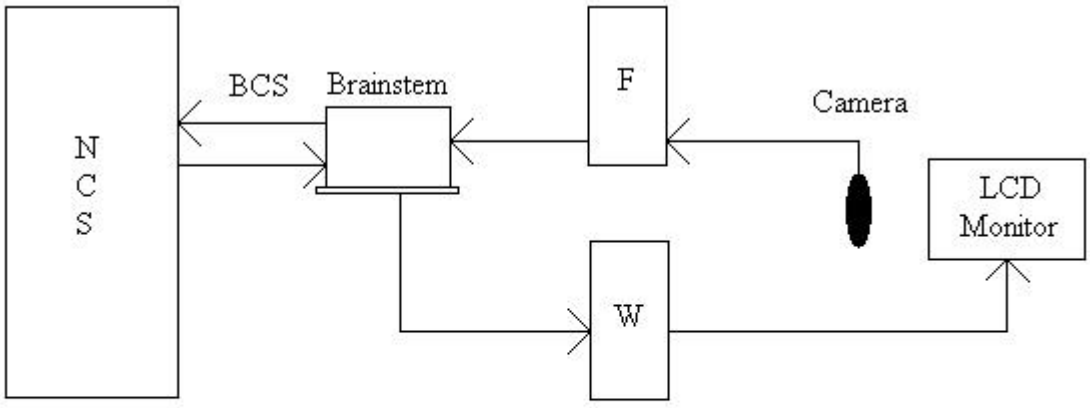
Apart from Cortex cluster, 3 PCs, a frame grabber card, an LCD monitor and a video camera were a part of the Virtual Social Robot (VSR) benchmark system. The system also used a driver for the camera, Webots [19], Universal Real-Time Behavior Interface (URBI) for Webots [17], Microsoft® Office® PowerPoint animation, NCS 5, BCS and Brainstem software applications. Brief details of the system setup (hardware and software) follow:

- The Cortex cluster was needed to execute NCS 5 and Brain Communication Server (BCS), also known as VoServer. The Opteron nodes of Cortex were used for the benchmarks. An AMD Opteron™ processor node is made up of two processors operating at 2.14 GHz with 4 GB RAM per node. The Cortex head node runs the Linux operating system with kernel release: 2.6.9-42.0.2.ELsmp. gcc version: 3.4.6 20060404 (Red Hat 3.4.6-3). MPICH version: mpich-1.2.7p1.
- Brainstem was running on an IBM® Thinkpad® X60, having Intel® Core™ Duo processor with 2 GB of RAM. It runs the Fedora Linux distribution, kernel release: 2.6.16-1.2096_FC5smp. gcc version: 4.1.0 20060304 (Red Hat 4.1.0-3).
- Another PC, an AMD® Opteron® Processor 250 (2.39 GHz) with 2 GB RAM and Microsoft® Windows® XP Professional Version 2002 Service Pack 2, was used to operate Webots and URBI for Webots software applications. This PC also had a PowerPoint file animation running on it. This animation has a blank slide initially. The next slide has animation of vertical black bars moving horizontally back and forth for 6 seconds. After the animation is over, a black screen is displayed till the user closes the PowerPoint slide show.
- A third PC (Intel® Pentium® 4 processor, 1500 MHz, 1 GB RAM) was needed for using the frame grabber card (via PCI interface). It runs the Ubuntu, kernel release: 2.6.20-15-generic.
- A Sony EVI-D70 color video camera was used to get the visual input stimulus for the VSR system.
- An LCD monitor was used to display the PowerPoint animation.

Except for the Cortex component, all other hardware and software in the system was present in the BCL.

4.1.2 Setup

Figure 4.1 gives a schematic setup of the system employed for the benchmarks. A brief description of the hardware and software components involved in the system was given in previous section. Figure 4.1 shows the benchmark system interaction.



NCS - Neo Cortical Simulator running on Cortex.
BCS - Brain Communication Server (VoServer) running on Cortex head node.
F - Frame grabber on a PC (PCI interface).
W - Webots, URBI for Webots and MS Office PowerPoint animation on a PC.

Figure 4.1 Schematic representation of the benchmark system setup.

The PC, indicated as W, is set up to display PowerPoint animation on the LCD Monitor in front of the camera. The frame grabber (indicated as F) captures images from camera and gives them as input stimulus to Brainstem. W also has Webots and URBI for Webots running on it, which simulates a virtual AIBO robot. Brainstem passes on this input stimulus via BCS to the NCS 5 neural network. The neural network processes the input stimulus and sends a report back to Brainstem, again with the help of BCS. These

report values are interpreted into one of the three pre-determined actions for virtual AIBO by Brainstem and sent to Webots on W. The virtual robot acts accordingly. This completes the VSR loop.

4.2 Procedure

The procedure for executing the benchmark tests is straight forward. It is as follows:

- Assuming that all necessary files are present and the system is configured correctly.
- Start the camera capture application such that images in form of JPEG files are being written to a location accessible by Brainstem.
- Start the Webots server and then the application itself. (URBI for Webots support is already built-in to Webots).
- Start the PowerPoint animation slide show. Note that initially just a blank (white screen) slide is displayed.
- Start the BCS.
- Now start the NCS 5 neural network simulation.
- Start Brainstem with GUI.
- This is the important part. Simultaneously press “enter” (or “spacebar”) to begin the slide show and “Start” button on Brainstem GUI to begin the simulation. This method is a bit crude and might induce errors of up till 500 ms. An automated approach to eliminate any manual error should be considered in the future. So

multiple simulations were done (about 5 per test) and a mean result was recorded to ensure consistency in the benchmarks.

4.3 Tests Design

Tests were designed keeping the hardware aspect in mind. The neural network had cell groups located in two layers in different columns. Each cell group consisted of 16 (or a multiple of 16) neurons, because the input stimulus coming from Gabor filters on Brainstem had only 16 values. Hence a single value was given to each neuron (in cases where the cell groups had total neurons as a multiple of 16, stimulus was duplicated for each set of 16 neurons). The total number of neurons in each cell group was kept same for all benchmark tests.

All simulations were 1.5 seconds long. Stimulus duration (via animation slide show) was 6 seconds. The tests were carried out with 2 and 32 cell group configuration, running on 1, 2, 4, 8, 16 and 32 CPUs when possible. The total count of neurons and CPUs in the tests was always doubled in the scenario of the next level. In some cases, the number of neurons was increased by up to 16 fold as it was known that intermediate results were similar to the previous scenario. Only one processor was used from each node, thus utilizing 32 nodes at most.

The main idea in the experiment was to record the time it takes for a meaningful behavior state transition for the virtual AIBO robot (viz. from watch state to angry state). This time was recorded to a text file by the Brainstem code. Corresponding System Time Response (STR) was also recorded from NCS 5 debug output. STR was the time it took

for NCS 5 to execute the neural network from first time tick to the last time tick of the simulation.

Layer 1 was configured to get input from Vertical Gabor filter and Layer 2 got input from Horizontal Gabor filter. Layer 1 was associated with the angry behavior and Layer 2 with happy behavior for the virtual AIBO. This was done by a piece of code in Brainstem.

The animation only had horizontal motion of vertical bars which were picked up by the Vertical Gabor filter and passed on to the cell groups in Layer 1. Thus the Vertical Gabor filter was always triggered by the PowerPoint animation. Cells in Layer 1 receiving this stimulus had spiking activity. This activity of cells was recorded as reports and sent to Brainstem.

The report given to Brainstem was from the first 16 cells from the first cell group in each of the layers. The cell group in a Layer which had maximum firing activity over a small window of time was considered to be the winning cell group for that window of time. To help Brainstem identify this winning cell group more easily, the spike shape was set to 25 time ticks for Layer 1 cell groups and 26 time ticks for Layer 2 cell groups, allowing a maximum firing rate of 40 and 38.46 spikes per neuron per second, respectively. Thus, we had cell groups in Layer 1 always firing at a higher rate than cell groups in Layer 2, which changed the state of the virtual AIBO from watch state to angry state.

This arrangement of Layer 1 cell groups always winning did not affect the timing aspect of benchmark results. We were not concerned about the intelligence in the

behavior of the system, as this was a benchmark experiment for measuring timing performance.

Four basic scenarios were simulated in six tests. Description of the tests is as follows:

- **Test 1**: This tested raw CPU power. There were no inter-connections between cells in the cell groups and hence no messages were passed.
- **Test 2**: This test analyzed the performance of Message Bus code of NCS which uses local memory (RAM) component of the system. For this, only self connections were made in cell groups with 10% probability. Messages were passed by the Message Bus on the same node via local memory.
- **Test 3**: Ethernet latency was tested here. Cell groups were now connected between each other with 10% probability. Self connections from Test 2 were removed. This made the Message Bus use MPI to pass messages across the Ethernet.
- **Test 4**: This test made use of both local memory and Ethernet and served to complete benchmark scenarios. Thus Test 4 combined Test 2 and Test 3 models. Cell groups had self connections and were also connected to each other at 10% probability.
- **Test 3 B**: Another set of Test 3 was performed, but with connections at 1%. This was to further analyze effect of number of connections and amount of message passing occurring in the network.
- **Test 4 B**: Similarly Test 4 was duplicated with 1% connections.

Chapter 5

Results

In this chapter, we will discuss results for our work. First we will take a look at final RAIN results and then look at results from the benchmarks.

5.1 RAIN

Continuing from where we left off at the end of Chapter 3, let us see the final results for RAIN experiments. Figure 5.1 gives the voltage tracings for the first 4 cells from cell

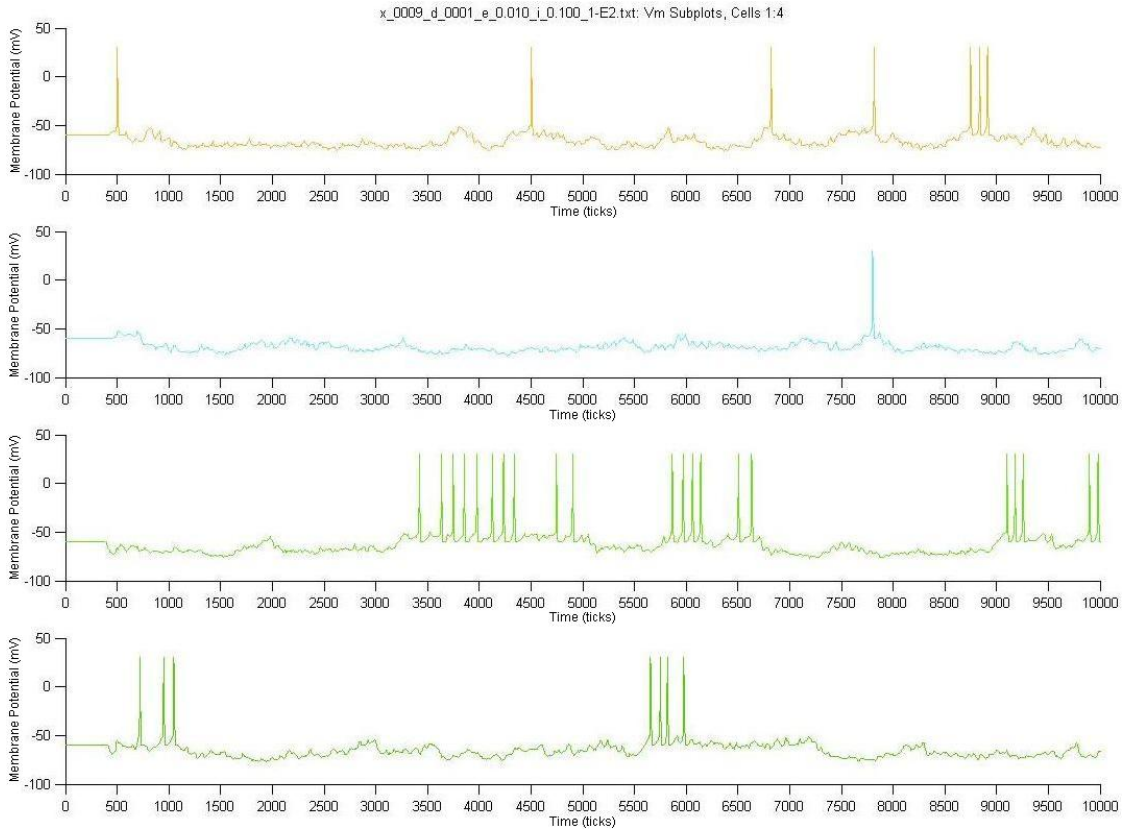


Figure 5.1 Membrane voltage tracings for first 4 cells in E2.

group E2. The spikes of all the cells are random, irregular and asynchronous indicating RAIN in the network.

Figure 5.2, 5.3 and 5.4 show the typical voltage report from cell groups E1, E2 and I respectively. We were able to obtain RAIN for both equal and binomial distribution connect algorithms. We have continued using the equal (forced) distribution algorithm.

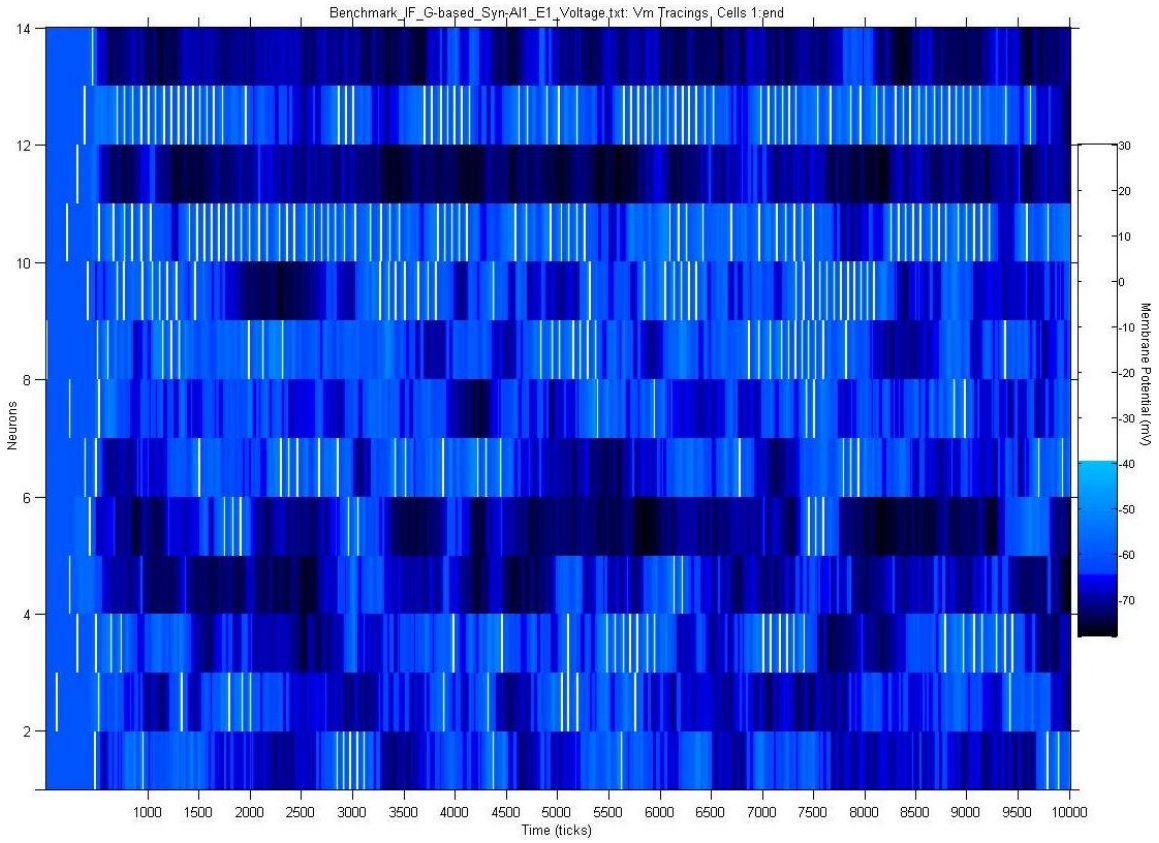


Figure 5.2 Voltage report plot of whole cell group E1 for final setup.

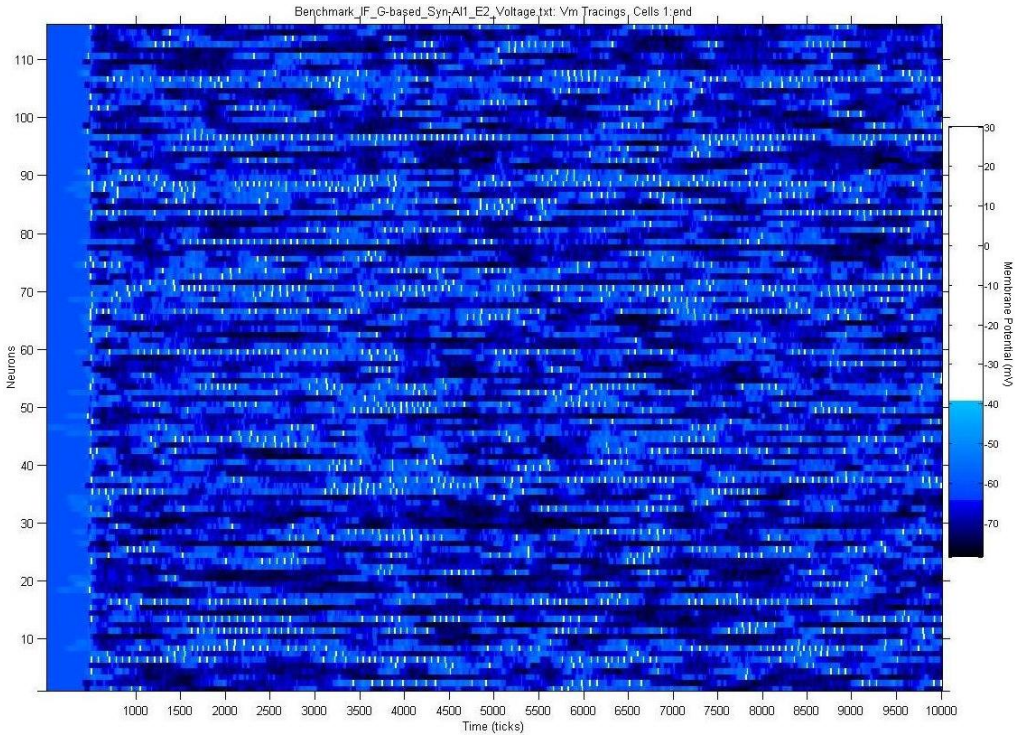


Figure 5.3 Voltage report plot of a few cells from cell group E2 for final setup.

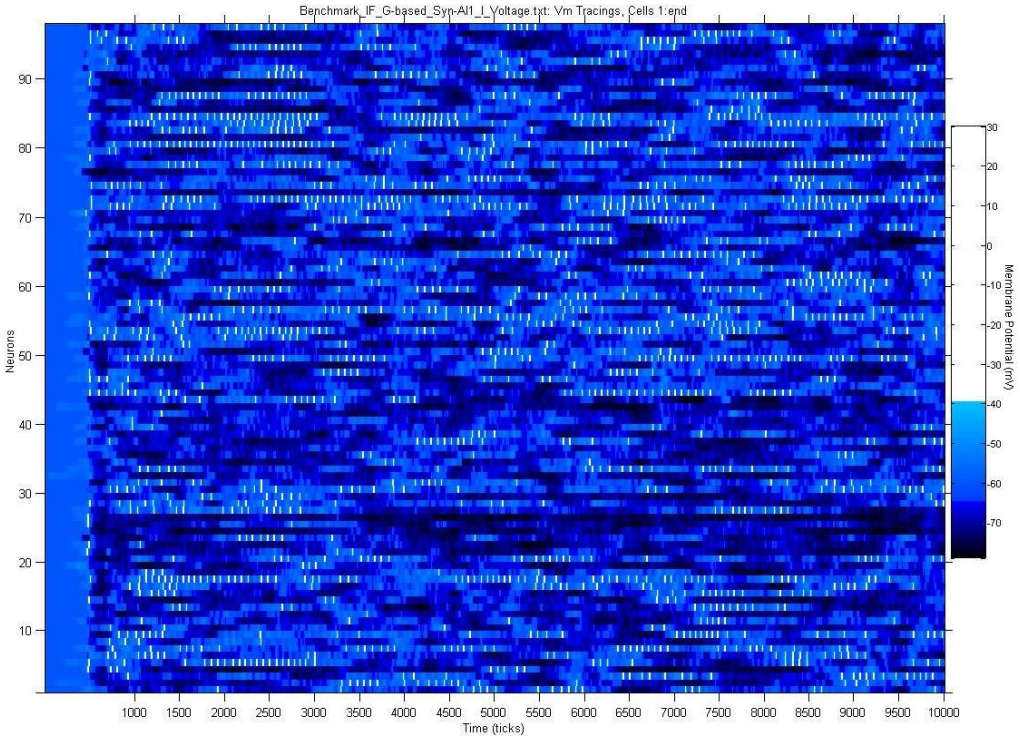


Figure 5.4 Voltage report plot of a subset of cell group I for final setup.

Figure 5.5 is from the Vogels and Abbott paper [68]. It shows the firing rates for numerous combinations of synaptic conductance. The x- and y-axes are inhibitory and excitatory conductance respectively, with the color indicating firing rates as represented in the legend color bar. The black square in the figure indicates the operating point that was chosen by the authors of the paper. The light blue triangular region represents RAIN activity region (about 12 to 25 Hz).

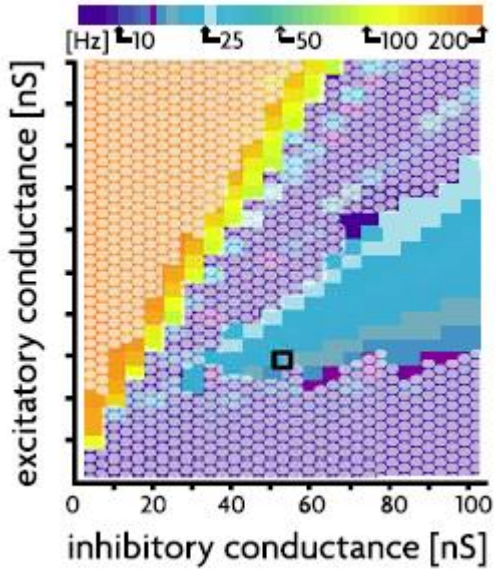


Figure 5.5 Firing rates for various combinations of excitatory and inhibitory synaptic conductance. [68]

We were also able to map out a similar triangular region with similar firing rates in our network, for different synaptic conductance, as shown in Vogels and Abbott paper in Figure 5.5. Corresponding plots have been shown in Figures 5.6, 5.7 and 5.8 for E1, E2 and I cell groups respectively. The axes are same as in Figure 5.5, with color indicating firing rates. Each combination of synaptic conductance represents a simulation.

Figure 5.9 shows different types of Network activities, including RAIN.

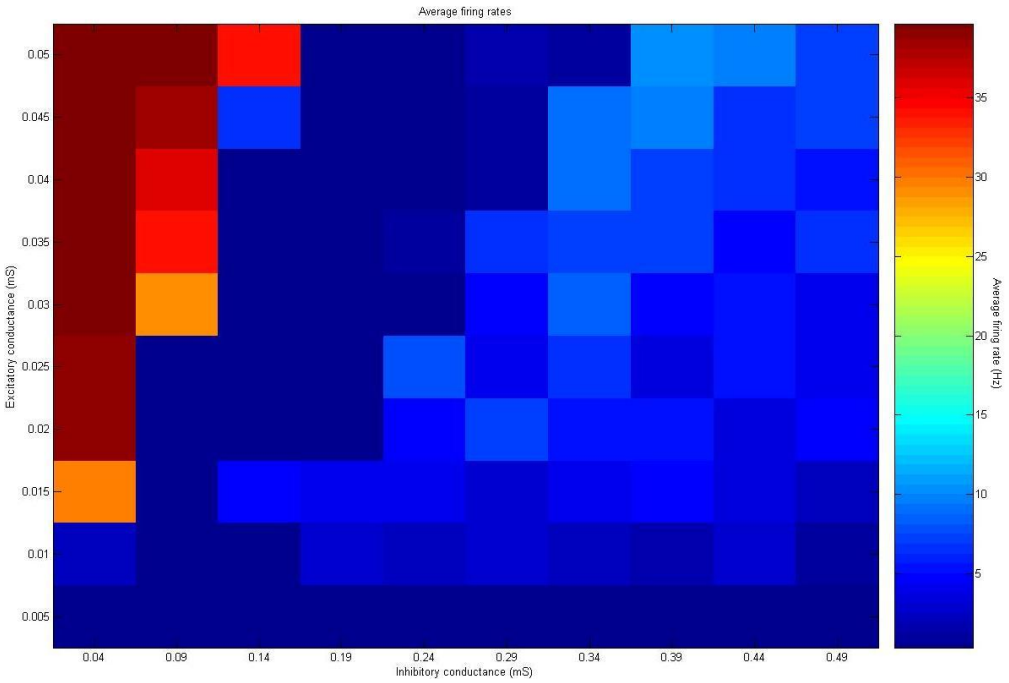


Figure 5.6 Firing rates for various combinations of excitatory and inhibitory synaptic conductance (E1 cell group).

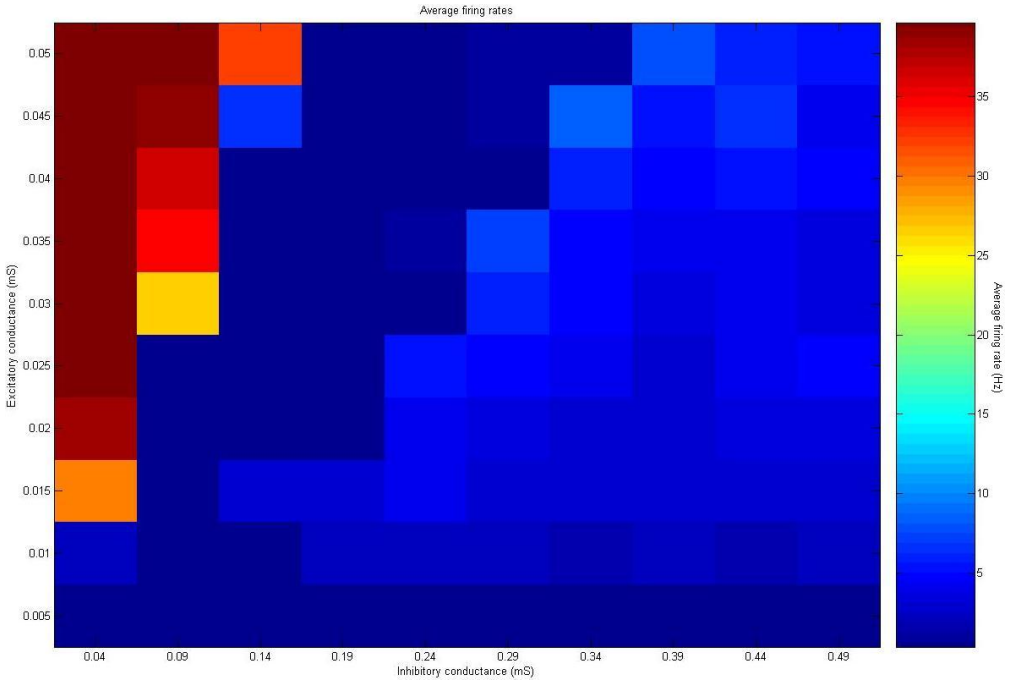


Figure 5.7 Firing rates for various combinations of excitatory and inhibitory synaptic conductance (E2 cell group).

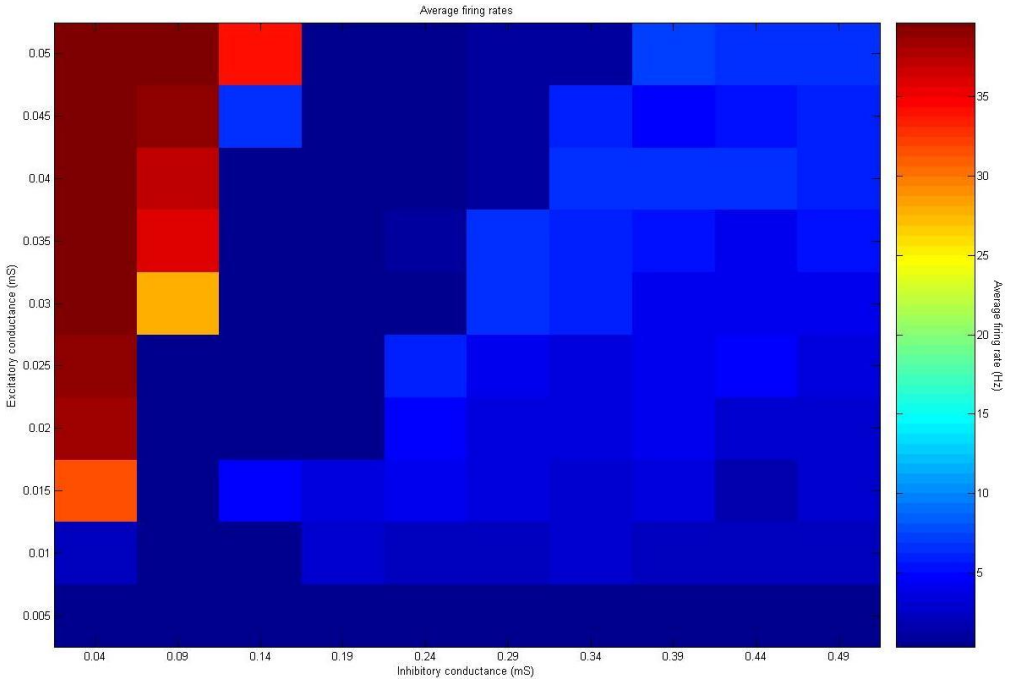


Figure 5.8 Firing rates for various combinations (I cell group).

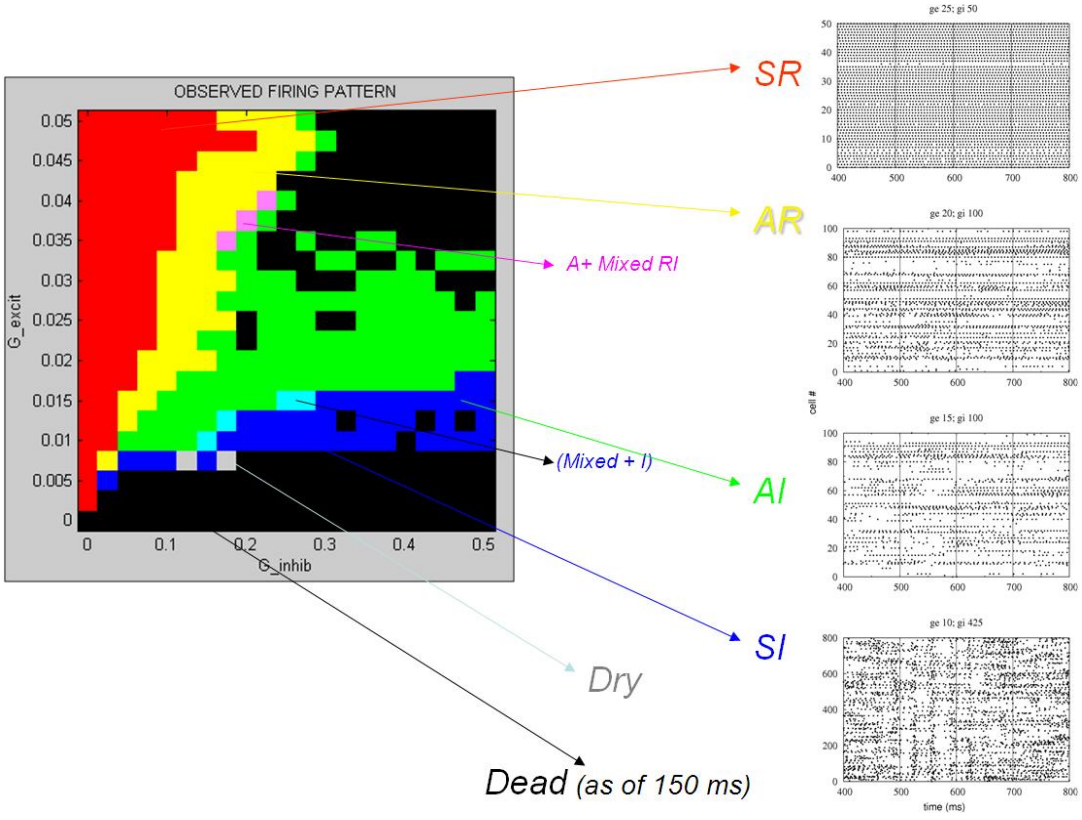


Figure 5.9 Types of network activity. [35]

In fact, the RAIN regime itself has other similar patterns. Those are Synchronous Regular (SR), Asynchronous Regular (AR), Asynchronous Irregular (AI) and Synchronous Irregular (SI). In addition to these behaviors, Figure 5.9 shows Asynchronous + mixed Regular-Irregular (A + Mixed RI), Mixed + Irregular (Mixed + I), Dry and Dead behaviors.

RAIN activity has proven to be fairly self sustained as seen in Figure 5.10. Here the simulation duration was 20 seconds and RAIN activity persisted till the end of simulation.

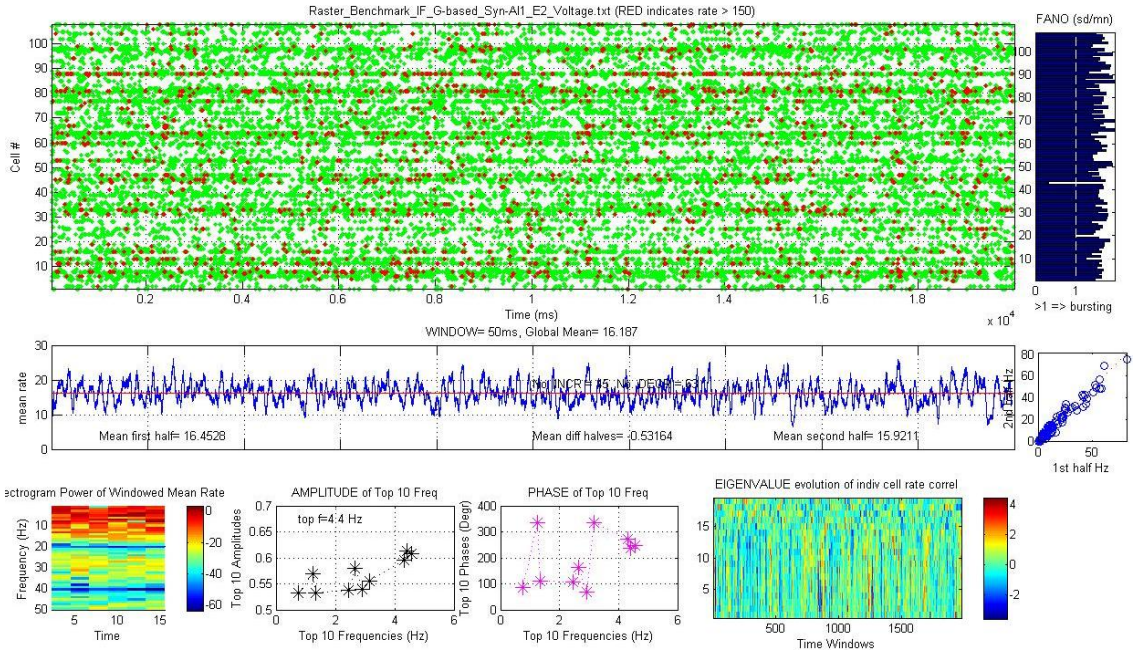


Figure 5.10 A plot from Analyze Raster MATLAB script for a subset of cells from E2 showing that RAIN activity is present through out a 20 second simulation.

The plot has been generated from one of the versions of analyze raster MATLAB script, developed in BCL. The major feature of the plot is the raster plot with spikes shown as green dots and high firing rates by red dots (spikes with very low inter-spike interval). Other windows are showing average firing rate, fan out (variability),

spectrogram power of windowed mean rate, amplitude of top 10 frequencies, phase of top 10 frequencies, and Eigen value correlation evolution of individual cells.

Experiments (inspired by [64]) to simulate sleep state (delta waves) brain activity and re-normalization of synaptic weights were also done with some success. These experiments used RAIN along with a modified network (E1, E2, E3, E4 and I, each having 800 cells) operating at **FSV** 1,000, negative ramp stimulus and RAIN pattern itself was used as a stimulus (taken from a standardized experiment earlier). Simulation duration was 15 seconds.

We saw in experiments that a RAIN network was more easily started (as seen in Figure 5.11) if a negative ramp stimulus was applied initially for about at least 25 ms. A

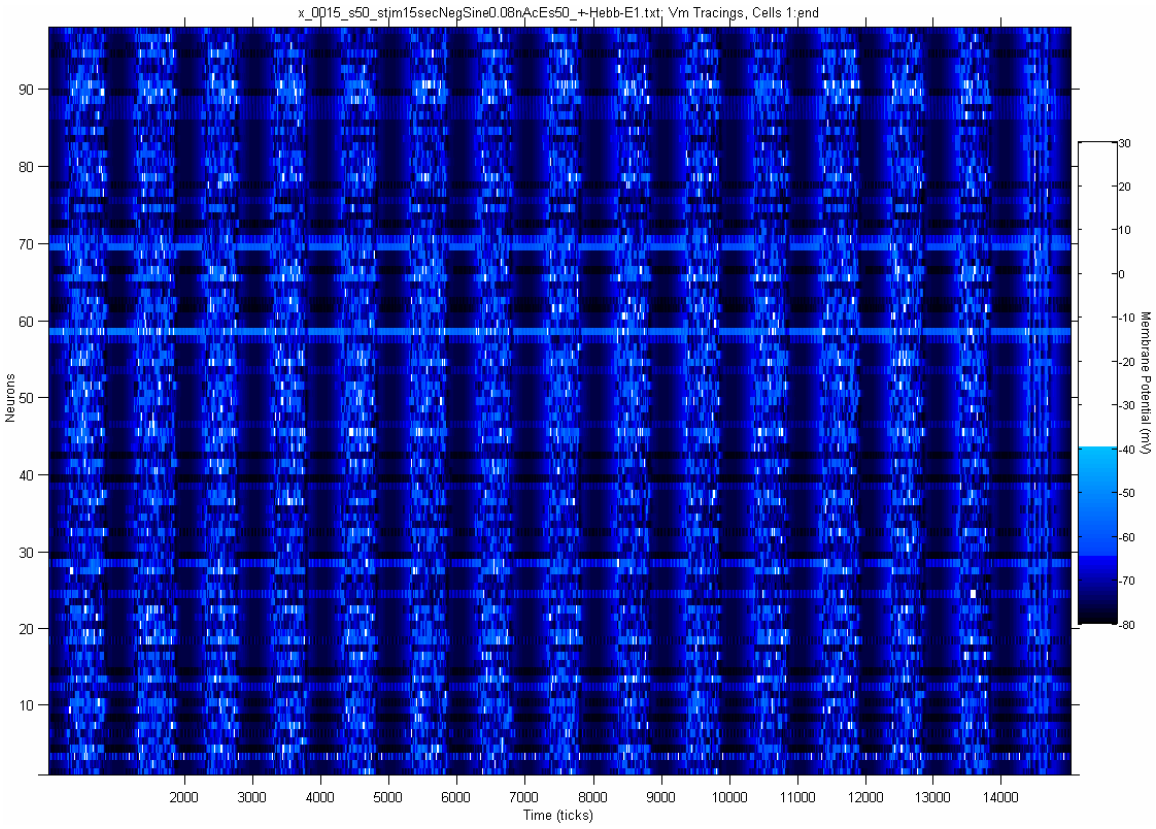


Figure 5.11 Slow wave oscillations in cell group E1. [35]

sine stimulus with different phases (in experiments) was used as the background stimulus for this RAIN network. STDP (positive and negative Hebbian learning) was also used.

Figure 5.11 shows the voltage report plot for cell group E1 with slow wave oscillations in network activity. Figure 5.12 depicts a downscaling in synaptic weights gradually over the duration of the simulation, before stabilizing. This represents a similar decrease of biological synaptic weights and thus simulating the sleep model suggested in the Tononi paper [64].

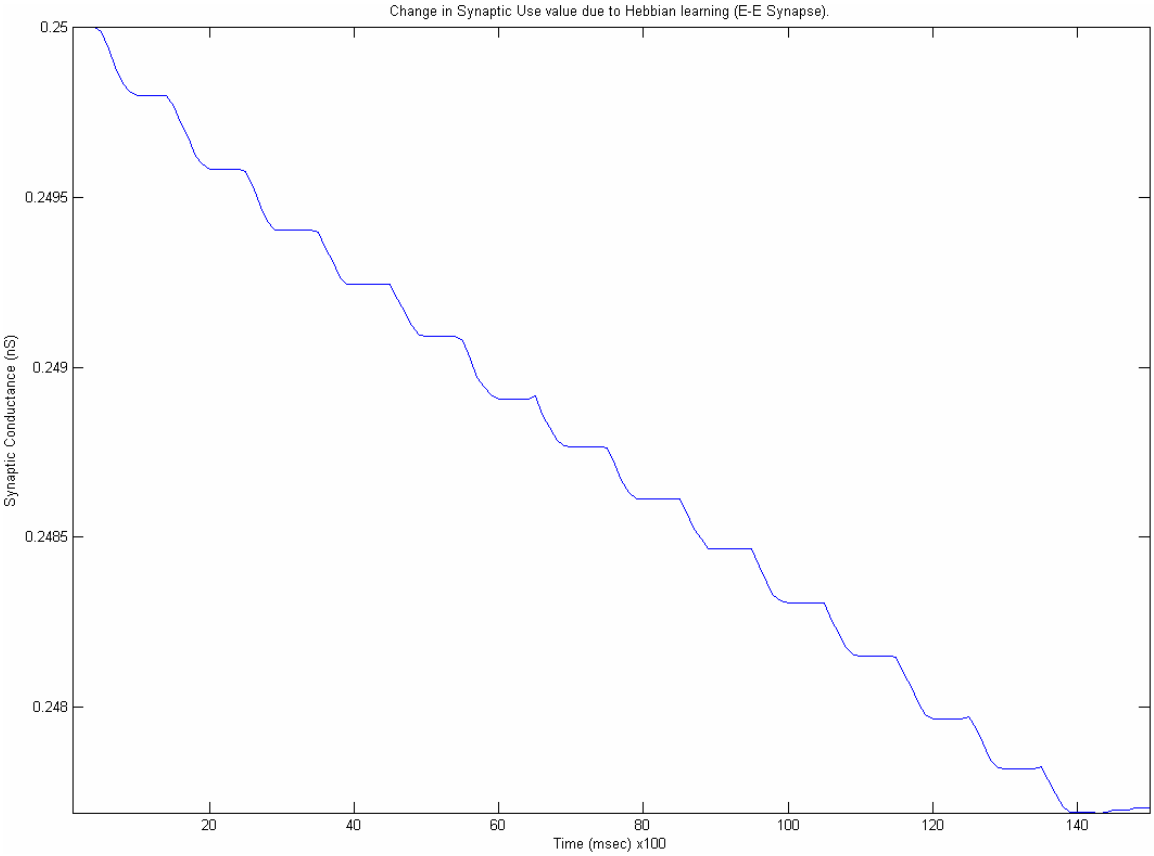


Figure 5.12 Re-normalization of synaptic weights due to Hebbian learning. [35]

Another set of experiments followed, which studied the least number of cells that were needed to be stimulated with RAIN stimulus and the smallest network able to sustain RAIN. Negative ramp stimulus and RAIN stimulus were used. The stimulus were applied

for 100 ms and given to only 25 E1 cells (Setup was same as mentioned previously, E1, E2, E3, E4 and I, each having 800 neurons) at **FSV** 1,000. This produced RAIN activity in the network. Also, the smallest network that was able to sustain RAIN for smaller periods of time was a total of 1000 neurons at **FSV** 1,000. This has important implications in having real-time applications for RAIN. Later on it was found that a model of network size 2,000 neurons, operating at **FSV** 10,000 was more robust than the 1,000 neuron model.

5.2 Benchmark Results

The resultant times were recorded for first meaningful transition in behavior of the virtual dog (angry, watchful or happy) indicated by a GUI on Brainstem (Behavioral Time Response). The response times were recorded in a file "timerespond.txt" by Brainstem code. NCS 5 execution times were recorded from debug messages by NCS 5 itself (System Time Response).

The values in the tables represent time in seconds. Two types of neural network models were used, one with 2 cell groups and another with 32 cell groups. The number of processors and-or cells are normally doubled while going on to the next level of test. In some cases where the results were remaining fairly constant, increments were more than double.

The number of processors range from 1 to 32. Values in bold indicate threshold after which the response was slower than real time (more than nearly 3 seconds). Corresponding system times are also in bold. CPUs represent the number of processors

that were used to run the brain model on. A value of "X" indicates a particular scenario was not necessary as it yields a similar result as in previous case or is beyond the range of our interest. A value of "NA" indicates that the resulting time was immeasurable.

5.2.1 Test 1

Behavioral Time Response (BTR):

Table 5.1 was split up into (a) and (b) because of space restrictions. Cell groups are depicted as follows. Consider the first column, 2 (16 x 1) in Table 5.1 (a). First number represents the number of cell groups in the model, here it is 2. First number in the bracket (here 16) is always the same in all cases. It represents the first basic set of 16 cells. Multiplication is shown by x. The second number in the bracket (here 1) is the number of sets of 16 cells that are present in the network. Thus total number of cells is calculated as 2 x 16 x 1 = 32 cells.

Table 5.1 (a) Results for 2 cell group neural network model.

CPUs	**Cell groups**			
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**	**2 (16 x 64)**
1	1.488	1.56	1.553	1.5
2	X	X	X	X

Table 5.1 (b) Results for 2 cell group neural network model (continued).

CPUs	**Cell groups**			
	2 (16 x 128)	**2 (16 x 512)**	**2 (16 x 1024)**	**2 (16 x 2048)**
1	1.497	1.715	**3.476**	6.403
2	X	X	1.78	**2.727**

Table 5.1 (b) shows that the maximum number of cells able to be simulated in real time for Test 1 using just a single processor, was 32768 (2^{15}). This is seen by bold value of 3.476 seconds in column 2 (16 x 1024) which gives 2 x 16 x 1024 = 32768 cells. Although, the thing to remember is that only half of the cells (Layer 1) here are spiking. Similarly 2 processors (2 CPUs) can simulate 65536 (2^{16}) cells, from 2 (16 x 2048) cell group column.

In Table 5.2 (a), the maximum number of cells that can run on 8 processors is 262144 (2^{18}) (calculated as 32 x 16 x 512), since the BTR was 3.084 seconds (near real time). In Table 5.2 (b), real time BTR for 32 cells was 3.136 seconds. This result was obtained by a 2^{20} cells model.

Table 5.2 (a) Results for 32 cell group neural network model.

CPUs	**Cell groups**			
	32 (16 x 64)	**32 (16 x 128)**	**32 (16 x 256)**	**32 (16 x 512)**
1	**3.148**	5.833	9.516	X
2	1.332	**2.963**	5.532	9.2
4	X	1.661	**3.099**	5.595
8	X	X	1.984	**3.084**
16	X	X	X	2.255
32	X	X	X	X

Table 5.2 (b) Results for 32 cell group neural network model (continued).

CPUs	**Cell groups**		
	32 (16 x 1024)	**32 (16 x 2048)**	**32 (16 x 4096)**
1	X	X	X
2	X	X	X
4	13.201	X	X
8	5.471	12.575	X
16	**4.265**	7.766	16.274
32	1.96	**3.136**	8.239

System Time Response (STR):

Tables 5.3 (a) and (b) give corresponding NCS system response for benchmark model of Test 1 involving 2 cell groups.

Table 5.3 (a) Results for 2 cell group neural network model.

CPUs	Cell groups			
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**	**2 (16 x 64)**
1	13.12259	13.22949	13.09861	13.212824
2	X	X	X	X

Table 5.3 (b) Results for 2 cell group neural network model (continued).

CPUs	Cell groups			
	2 (16 x 128)	**2 (16 x 512)**	**2 (16 x 1024)**	**2 (16 x 2048)**
1	13.109415	16.211114	**28.529475**	53.849343
2	X	X	16.544322	**29.765697**

Tables 5.4 (a) and (b) give STR for 32 cell group networks. Real time STR for an 8 processors simulation is 32.42 seconds shown as a bold value in column 32 (16 x 512) in Table 5.4 (a).

Table 5.4 (a) Results for 32 cell group neural network model.

CPUs	Cell groups			
	32 (16 x 64)	**32 (16 x 128)**	**32 (16 x 256)**	**32 (16 x 512)**
1	**27.03646**	50.714999	95.175041	X
2	14.572991	**25.910521**	48.37169	93.730504
4	X	16.112949	**32.759438**	63.190934
8	X	X	17.76706	**32.42442**
16	X	X	X	22.061521
32	X	X	X	X

Table 5.4 (b) Results for 32 cell group neural network model (continued).

CPUs	Cell groups		
	32 (16 x 1024)	**32 (16 x 2048)**	**32 (16 x 4096)**
1	X	X	X
2	X	X	X
4	125.722244	X	X
8	60.879575	119.477107	X
16	**39.930199**	80.739848	149.101989
32	17.552403	**32.711633**	63.289575

Similar test setup and design was used to simulate other tests. The exception being the way the simulations were allocated on Cortex cluster to test local memory (Test 2), Ethernet latency (Test 3) and both (Test 4).

5.2.2 Test 2

Behavioral Time Response (BTR):

As seen from Tables 5.5 (a) and (b), the BTR for Test 2 are much higher than seen in Test 1 for the same network setup.

Table 5.5 (a) Results for 2 cell group neural network model.

CPUs	Cell groups		
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**
1	1.5	1.547	1.54
2	1.526	X	X

Table 5.5 (b) Results for 2 cell group neural network model (continued).

CPUs	Cell groups		
	2 (16 x 64)	**2 (16 x 128)**	**2 (16 x 256)**
1	2.215	**3.828**	14.605
2	X	3.894	14.487

For example, maximum number of cells that could be simulated in real time for Test 2 were 4096 (2^{12}) compared to 32768 (2^{15}) cells in Test 1.

Similar to Table 5.5, Table 5.6 also has BTR higher than Test 1. 2^{14} cells were simulated in real time on 1 processor as indicated by BTR value of 3.773 seconds.

Table 5.6 (a) Results for 32 cell group neural network model.

CPUs	Cell groups		
	32 (16 x 8)	**32 (16 x 16)**	**32 (16 x 32)**
1	1.484	1.772	**3.773**
2	X	X	3.703
4	X	X	2.869
8	X	X	1.959
16	X	X	1.575
32	X	X	1.555

Table 5.6 (b) Results for 32 cell group neural network model (continued).

CPUs	Cell groups		
	32 (16 x 64)	**32 (16 x 128)**	**32 (16 x 256)**
1	7.882	X	X
2	10.117	X	X
4	5.316	15.34	X
8	**3.861**	11.283	X
16	2.282	5.657	19.87
32	2.223	**4.515**	10.693

System Time Response (STR):

Tables 5.7 (a) and (b) give STR for 2 cell group models from in Test 2.

Table 5.7 (a) Results for 2 cell group neural network model.

CPUs	Cell groups		
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**
1	13.09803	13.28663	14.30893
2	13.11258	X	X

Table 5.7 (b) Results for 2 cell group neural network model (continued).

CPUs	Cell groups		
	2 (16 x 64)	**2 (16 x 128)**	**2 (16 x 256)**
1	22.63773	**65.395473**	319.41792
2	X	65.818943	319.14783

Tables 5.8 (a) and (b) give STR result for 32 cell group model. Corresponding STR for 1 processor was 57.95 seconds.

Table 5.8 (a) Results for 32 cell group neural network model.

CPUs	Cell groups		
	32 (16 x 8)	**32 (16 x 16)**	**32 (16 x 32)**
1	13.670199	21.001584	**57.946136**
2	X	X	57.515205
4	X	X	34.955642
8	X	X	24.180679
16	X	X	19.032793
32	X	X	17.029839

Table 5.8 (b) Results for 32 cell group neural network model (continued).

CPUs	Cell groups		
	32 (16 x 64)	**32 (16 x 128)**	**32 (16 x 256)**
1	217.13195	X	X
2	208.36749	X	X
4	111.24735	461.188195	X
8	**61.347211**	235.667336	X
16	37.932387	124.895976	641.814124
32	25.951538	**69.359208**	331.443202

5.2.3 Test 3

A) Connections between cell groups are at 10%.

Behavioral Time Response (BTR):

The first part of Test 3 had cell groups connecting at a probability of 10% with other cell groups (across connections). Tables 5.9 and 5.10 represent results for 2 and 32 cell group brain models for Test 3 (part A). From Table 5.10, we see that as we go from one model to the next, the number of processors required in maintaining real time RTB quadruples instead of doubling as was the case in Test 1.

Table 5.9 Results for 2 cell group neural network model.

CPUs	**Cell groups**				
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**	**2 (16 x 64)**	**2 (16 x 128)**
1	X	X	X	X	X
2	1.641	1.712	1.81	**3.397**	13.525

Table 5.10 Results for 32 cell group neural network model.

CPUs	**Cell groups**			
	32 (16 x 4)	**32 (16 x 8)**	**32 (16 x 16)**	**32 (16 x 32)**
1	X	X	X	X
2	**3.532**	13.811	X	X
4	2.18	7.255	NA	X
8	1.884	**4.129**	16.903	X
16	X	2.652	9.759	X
32	X	1.792	**5.732**	NA

System Time Response (STR):

Table 5.11 and 5.12 portray results for 2 and 32 cell group models in Test 3 (part A). The number of cells simulated in real time using a single processor was 2048 (2^{11}). This is half of the number of cells from Test 2 and much lesser than in Test 1.

Table 5.11 Results for 2 cell group neural network model.

CPUs	Cell groups				
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**	**2 (16 x 64)**	**2 (16 x 128)**
1	X	X	X	X	X
2	13.12612	13.21849	16.17487	**34.92781**	125.11506

Table 5.12 Results for 32 cell group neural network model.

CPUs	Cell groups			
	32 (16 x 4)	**32 (16 x 8)**	**32 (16 x 16)**	**32 (16 x 32)**
1	X	X	X	X
2	**34.422668**	125.05567	X	X
4	22.45127	64.848998	303.018165	X
8	16.948477	**37.863951**	145.126107	X
16	X	25.423113	78.529517	X
32	X	18.61824	**45.640834**	175.67498

B) Connections between cell groups are at 1%.

Behavioral Time Response (BTR):

Part B of Test 3 had cell groups connecting across at 1% with other cell groups. Table 5.13 and 5.14 depict results for 2 and 32 cell group models in Test 3 (part B).

Table 5.13 (a) Results for 2 cell group neural network model.

CPUs	Cell groups			
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**	**2 (16 x 64)**
1	X	X	X	X
2	1.525	1.548	1.784	1.735

Table 5.13 (b) Results for 2 cell group neural network model (continued).

CPUs	Cell groups		
	2 (16 x 128)	**2 (16 x 256)**	**2 (16 x 512)**
1	X	X	X
2	2.07	**6.841**	NA

The number of cells simulated in real time using 2 processors was 8192 (2^{13}). Reducing connection probability increases the capacity of processors to simulate larger models, because message passing has reduced due to fewer connections, as expected.

Table 5.14 Results for 32 cell group neural network model.

CPUs	Cell groups			
	32 (16 x 8)	**32 (16 x 16)**	**32 (16 x 32)**	**32 (16 x 64)**
1	X	X	X	X
2	2.226	6.184	X	X
4	1.82	**3.36**	13.39	X
8	1.7	1.99	7.024	NA
16	X	1.832	**4.19**	NA
32	X	1.684	2.551	8.976

System Time Response (STR):

Again tables were needed to be broken up into (a) and (b) to manage space as seen in Tables 5.15 (a) and (b).

Table 5.15 (a) Results for 2 cell group neural network model.

CPUs	Cell groups			
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**	**2 (16 x 64)**
1	X	X	X	X
2	13.09831	13.10969	13.26272	13.22319

Table 5.15 (b) Results for 2 cell group neural network model (continued).

CPUs	Cell groups		
	2 (16 x 128)	**2 (16 x 256)**	**2 (16 x 512)**
1	X	X	X
2	20.005084	**56.4378**	225.97363

Table 5.16 gives STR results for 32 cell group models. The number of cells simulated in real time for Test 3 (part B) is approximately three times more than that of Test 3 (part A).

Table 5.16 Results for 32 cell group neural network model.

CPUs	Cell groups			
	32 (16 x 8)	**32 (16 x 16)**	**32 (16 x 32)**	**32 (16 x 64)**
1	X	X	X	X
2	19.613465	53.431618	X	X
4	14.920719	**31.402475**	107.582508	X
8	13.669482	21.319482	57.376585	243.18132
16	X	16.233758	**34.713016**	123.02606
32	X	13.850684	23.16983	65.592769

5.2.4 Test 4

A) Connections between cell groups at 10% and self connections in cell groups at 10%. Behavioral Time Response (BTR):

Tables 5.17 and 5.18 represent results for 2 and 32 cell group models connected at 10% probability in Test 4 (part A). From Table 5.18 we see that, as the total amount of cells in cell groups inside the neural network were doubled, the number if processors were needed to be increased four-fold to keep the BTR response id the system in real-time (around 3 seconds).

Table 5.17 Results for 2 cell group neural network model.

CPUs	Cell groups				
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**	**2 (16 x 64)**	**2 (16 x 128)**
1	X	X	X	X	X
2	1.631	1.735	2.114	**5.755**	NA

Table 5.18 Results for 32 cell group neural network model.

CPUs	Cell groups			
	32 (16 x 4)	**32 (16 x 8)**	**32 (16 x 16)**	**32 (16 x 32)**
1	X	X	X	X
2	**3.784**	14.316	X	X
4	2.213	7.444	NA	X
8	1.983	**4.209**	NA	X
16	X	2.728	10.636	X
32	X	1.895	**6.915**	NA

System Time Response (STR):

From Tables 5.19 and 5.20, we see the same trend of quadrupling number of processors for next level of model to be kept in real time, as was seen in Test 3 and BTR results of Test 4.

Table 5.19 Results for 2 cell group neural network model.

CPUs	Cell groups				
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**	**2 (16 x 64)**	**2 (16 x 128)**
1	X	X	X	X	X
2	13.15775	13.71662	21.43926	**58.89992**	270.03817

Table 5.20 Results for 32 cell group neural network model.

CPUs	Cell groups			
	32 (16 x 4)	**32 (16 x 8)**	**32 (16 x 16)**	**32 (16 x 32)**
1	X	X	X	X
2	**35.925011**	131.80859	X	X
4	23.134739	67.815451	323.774671	X
8	17.109711	**38.952764**	154.702883	X
16	X	25.849791	87.555658	X
32	X	19.160225	**54.862979**	245.92971

B) Connections between cell groups at 1% and self connections in cell groups at 10%.

Behavioral Time Response (BTR):

Tables 5.21 and 5.22 represent results for 2 and 32 cell group models connected at 1% probability between cell groups and 10% for self connections in Test 4 (part B).

Table 5.21 Results for 2 cell group neural network model.

CPUs	**Cell groups**				
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**	**2 (16 x 64)**	**2 (16 x 128)**
1	X	X	X	X	X
2	1.528	1.62	1.813	**3.156**	13.055

Table 5.22 Results for 32 cell group neural network model.

CPUs	**Cell groups**				
	32 (16 x 4)	**32 (16 x 8)**	**32 (16 x 16)**	**32 (16 x 32)**	**32 (16 x 64)**
1	X	X	X	X	X
2	1.557	2.188	8.144	NA	X
4	1.563	1.893	**4.244**	17.817	X
8	X	1.678	2.444	9.621	NA
16	X	X	1.864	6.223	NA
32	X	X	X	**4.299**	16.573

System Time Response (STR):

Contrary to results from Test 3 (part A and B), the difference in performance for Test 4 (part A and B) is not that large, as seen from Tables 5.23 and 5.24.

Table 5.23 Results for 2 cell group neural network model.

CPUs	**Cell groups**				
	2 (16 x 1)	**2 (16 x 16)**	**2 (16 x 32)**	**2 (16 x 64)**	**2 (16 x 128)**
1	X	X	X	X	X
2	1.528	1.62	1.813	**3.156**	13.055

Table 5.24 Results for 32 cell group neural network model.

CPUs	Cell groups				
	32 (16 x 4)	**32 (16 x 8)**	**32 (16 x 16)**	**32 (16 x 32)**	**32 (16 x 64)**
1	X	X	X	X	X
2	13.724195	24.441637	75.432045	343.89577	X
4	13.196295	17.401348	**41.606433**	162.96342	X
8	X	14.309163	26.491229	83.442747	382.899589
16	X	X	19.859836	54.70997	217.900018
32	X	X	X	**37.58362**	129.584271

Although across connections were dropped from 10% to 1%, the reason is the additional overhead of passing messages on local memory due to 10% self connections.

5.2.5 Graphs

Effect of Message Bus Code Due to Spikes

In this analysis we take a look at the effect of delay due to spikes occurring in the neural network. We have calculated the total number of spikes for a network during simulation. The System Time Response (STR) was then taken in terms of μ-seconds and divided by total number of spikes to give "Microsecond / Spike". This value indicates how much computation power was spent on two major code portions, viz. message bus code and code dealing with maintaining state (like compartment equations) of the neural network. A large value indicates that processors were mostly busy executing the message passing code and a small value, say 1, indicates that the processors spent more time on executing compartment state code.

Log plots (cells vs. delay) were employed to better demonstrate results, as the tests mostly involved doubling of cells and processors. An important point to note here is that true results are seen only when a neural network is spiking near its maximal spiking rate.

Figure 5.13 represents a comparison between Tests 1 and 2 done on a single processor with only Layer 1 spiking. Initially Test 2 seems efficient, but after 1024 cells mark, Test 2 processor begins to spend more time on Message Bus code for message passing. This happens because, neural network in Test 2 starts firing maximally (40 Hz) only after 1024 cells mark showing the true result. Note that Test 1 curve is a straight line with slope 1. This means that a processor is utilized optimally for both pieces of code, as number of cells increases (also referring to intermediate results).

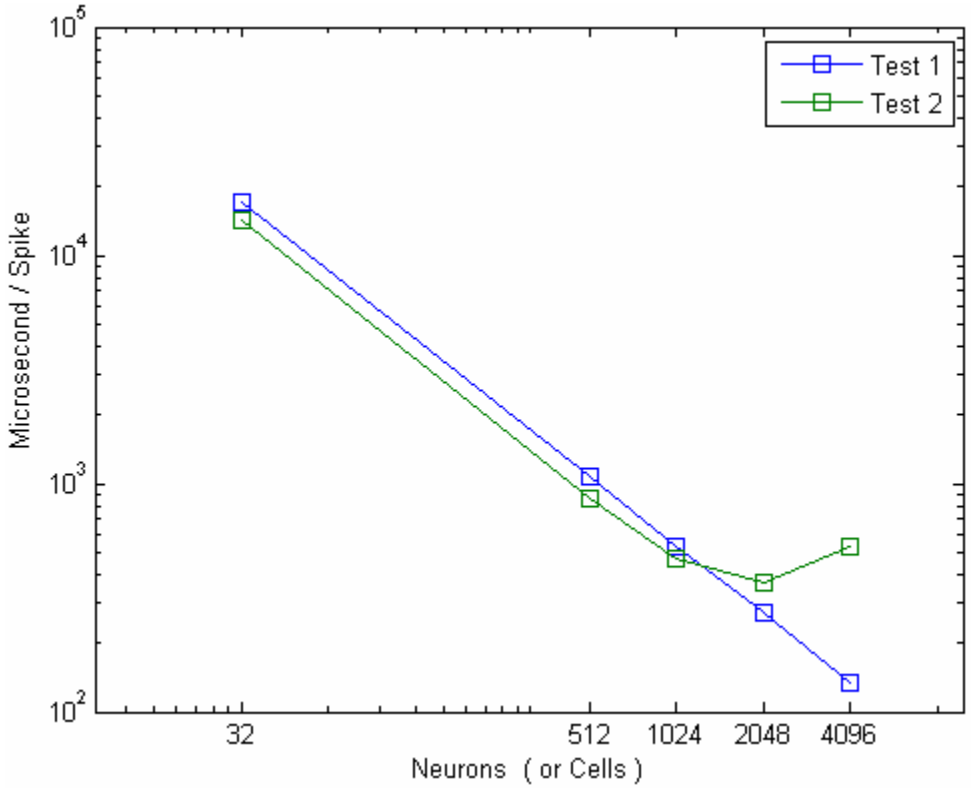


Figure 5.13 Test 1 vs. Test 2.

In Figure 5.14, we see a comparison between Test 3 and Test 4 performed on two processors with both Layers spiking. Again, we see that for first mark (32 cells) Test 4 is not spiking much, thereby having a low Microsecond / Spike value. For remaining points, as expected, processors spent more time on Message Bus code for Test 4 than Test 3. Because Test 4 has more messages being passed due to self connections in cell groups which are not present in Test 3. Another thing to note is that as number of cells keeps doubling, Microsecond / Spike value increases non-linearly in models where Ethernet (or local memory) is involved.

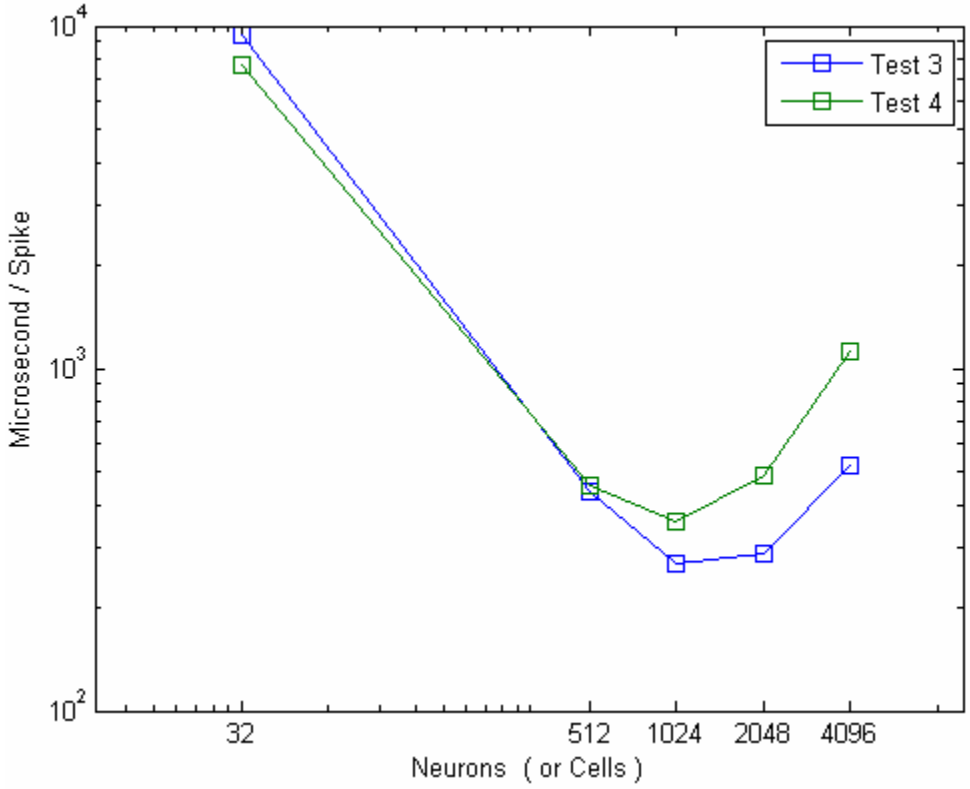


Figure 5.14 Test 3 vs. Test 4.

Figure 5.15 compares Test 3 at 10% and 1% connections on two processors with both layers spiking. Again we see a similar trend here. As the 1% connected neural network starts spiking maximally (after mark of 512 cells), it has small Microsecond / Spike value

than 10% connected model. In fact, when this value is increasing in 10% model, it goes on reducing slowly in 1% model.

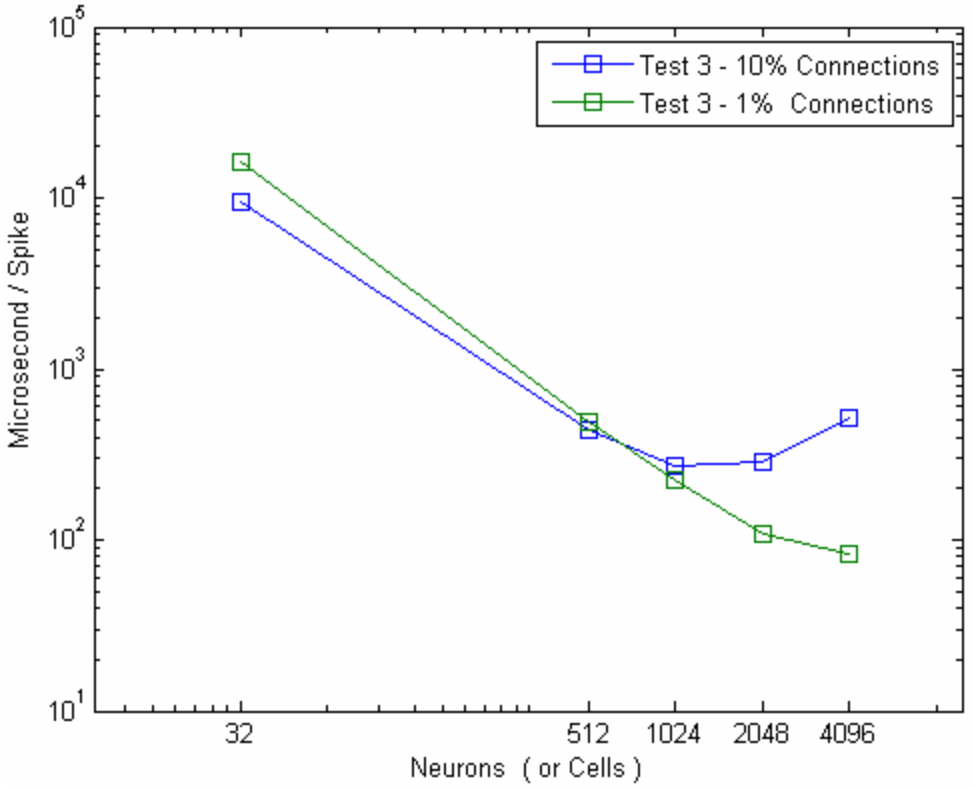


Figure 5.15 Test 3 - 10% vs. Test 3 - 1%.

A similar trend is seen in Figure 5.16, Test 4 at 10% and 1% connections on two processors with both layers spiking. The difference is that the Microsecond / Spike value in 1% model also goes on increasing, as in case of 10%, albeit lesser than 10%. As expected, reducing number of connections yielded in lesser time being spent on Message Bus code, when the neural networks were spiking near their maximum spiking rate.

From the plots we see that having spikes causes message passing in the NCS 5 Message Bus, thus making the processor spend more time on that piece of code. Large simulations not involving Message Bus (Test 1) have greater efficiency, by almost an order of magnitude compared to local memory and Ethernet simulations (Test 2, 3 and 4).

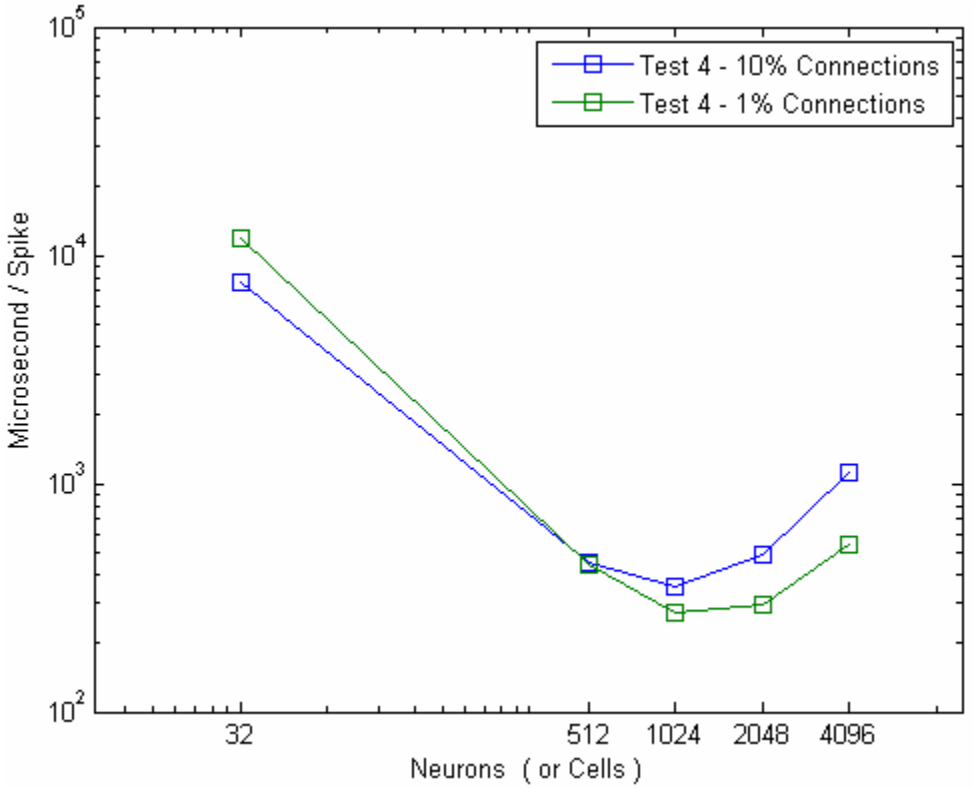


Figure 5.16 Test 4 - 10% vs. Test 4 - 1%.

Looking at Test 3 and Test 4 (local memory vs. Ethernet), for 32 (16 x 8) cells running on 16 nodes the behavioral time responses are comparable (2.65 and 2.73 seconds respectively). We get values of 105 and 107 µ-s/spike, a difference of 2 µ-s/spike. So we can roughly say that Ethernet is 2/100th or 1/50th or 50 times slower than local memory.

Real Time Behavioral Response of the System

We now compare all the basic four tests for real time behavioral responsiveness. A Log plot (processors vs. cells) is used again for previously mentioned reasons. The closest cell and processor combinations were noted where the system showed that it was just or slightly above real time. Selecting points near or faster than real time reduced the number of plot-points, especially in case of Test 3 and Test 4.

Figure 5.17 shows comparison of real time behavioral response. This indicates that processor cache is the fastest medium for message passing and copes best with increasing network size. Doubling the cells and processors achieves real time responsiveness of the system in Test 1. Test 3 and 4 had the same data points and hence are shown by the same line.

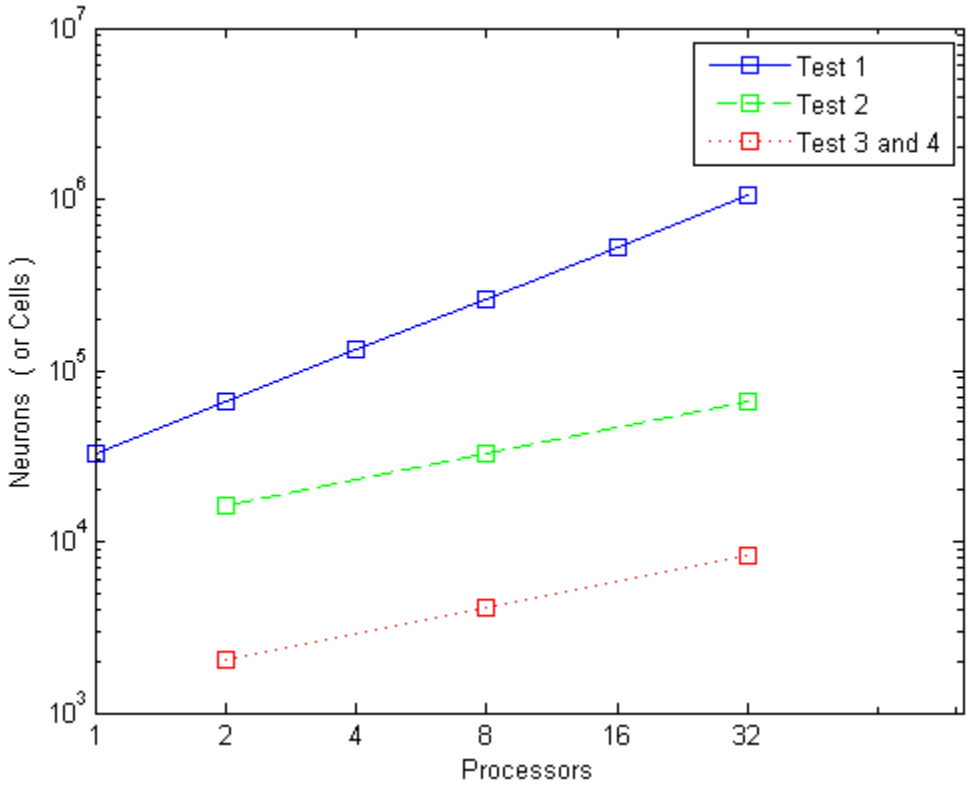


Figure 5.17 Comparison of real time behavioral response.

Test 1 is a straight line with slope 1. Tests 2, 3 and 4 have their plot as a straight line but with slope not equal to 1. This means that, simply doubling processors for an increase in number of cells, as in Test 1, does not maintain real time system response. Number of processors was in fact increased four (or even eight) times for a two-fold increase in cell count, to maintain real time behavior. The point that Ethernet is slower by almost an

order of magnitude (in terms of network size) than local memory is seen by comparing Test 2 and Test 3 and 4 lines. Number of cells simulated by Test 2 is almost 10 times more.

Similar scenario arises when we compare Test 1 and Test 2 plots. Processor cache is faster than local memory for message passing. Number of cells simulated by Test 1 is about 10 times more than Test 2. This is ever so true as the network size increases.

Other Important Observations

- The major time consuming factor in NCS 5 is message passing due to spikes.
- Organization of the number of cells in connected cell groups is important. For example, in Test 2, for same number on cells (4096) executing on same number of processors (1) yielded a significant difference in behavioral (2.344 seconds) and system response (51.7 seconds). The difference in these setups is that the first case had 2 cell groups and the latter had 32 cell groups. So the number of synapses per processor is different in the two cases. Each processor has a limit to the number of synapses it can compute or update in real time. First case has 838,860 synapses (2 connects x ((16 cells x 128 cells)^2 x 10%)) and in the second case there are 52,416 synapses (32 connects x ((16 cells x 8 cells)^2 x 10%)). This is an important point to be kept in mind when designing neural networks with NCS 5.
- In case of massive networks, the delay in processing was so large that the BCS-Brainstem buffers were full almost immediately, as NCS was not ready to accept incoming stimulus. Due to this, input stimulus (6 seconds duration) coming in after the buffers were full, was lost. Thus there was no behavioral response in the

system as input stimulus was missed and the reason for having “NA” in results for few experiments.

- Another system issue is the report files. When a model has reports, they are stored in files. Writing to a file requires disk I/O, which is slower than fast Ethernet connection. So in case of large report files, the simulation slows down slightly because NCS 5 needs to write to disk before it can proceed to simulate the next time tick. This was taken into consideration while performing benchmark tests by recording results from the simulations in which reports were turned off. To find total number of spikes, a duplicate simulation was done with reports on.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Models of neural network demonstrating RAIN behavior were developed successfully [24]. After detailed analysis and experimentation, the key factor for generation of RAIN activity in our neural networks was found to be the self connections in the inhibitory cell group. These self connections in inhibitory cells act as a meta-parameter which controls the amount of inhibition applied to excitatory as well as inhibitory neurons in the network. This is also representative of a feedback mechanism for the inhibition system.

RAIN has the promise of being the stepping stone for a major breakthrough principle in simulating mammalian (especially human) brain activity and behavior, sought after by those in computational neuroscience and other fields. As mentioned in Chapter 5, RAIN was successfully used to give a demonstration of slow wave deep sleep (delta waves). In conjunction with Hebbian learning, re-normalization of synaptic weights was modeled as an effect of delta wave oscillations found in biology [35].

Other experiments like the paradigm of “winner takes most” were done, but with small degree of success.

Benchmark test results have shown that the major bottleneck is the messages (spikes) that are sent to cell groups. Irrespective of hardware medium, CPU cache, local memory or Ethernet used, message transmission is the single most computationally expensive

feature of NCS 5. With respect to hardware, as expected Ethernet is the slowest, followed by local memory and then CPU cache.

Upgrading of hardware raises a basic question. Do we need more computation power in form of large number of CPUs or a limited computation power with a high speed interface? Obviously this is a tradeoff which must be achieved depending upon future requirements. If large numbers of CPU are needed, they can be used along with Ethernet or similar relatively cheap interface, sacrificing performance. Otherwise money could be invested in a shared memory system, where number of CPUs is sufficient and faster communication happens through shared memory.

Another alternative is to get few nodes with latest multi-core processors (for example 32 processors per node) which would be connected with an InfiniBand network interface. This interface has speeds ranging from 2 Gbit/s to 96 Gbit/s in theory.

A fourth alternative is to look at new innovative architectures, like Lightfleet interconnect machines [11].

Neural modeling using computational devices has an important role to play in the future. To understand and unravel the mysteries of the brain and theory of cortical computation there is a need of tight combination between theoretical and experimental results [26]. Simulations along with the experiments are being successfully used by computational biologists to understand and predict the quantitative behavior of complex systems [67].

6.2 Future Work

Many experiments based on the RAIN concepts have happened with small amounts of success. Re-evaluating them might prove useful to further them or sprout new ideas off of them. Simulating sleep and its effects with the help of RAIN need to be investigated further. Application of negative stimulus and use of RAIN itself as the driving stimulation needs to be studied in great detail. Models depicting “winner takes all” and “winner takes most” behavior in single and multiple networks, using RAIN, need to be modeled accurately.

Research is currently underway in the BCL to study pattern recognition in audio stimulus with help of RAIN and learning. Autism models will soon be developed using RAIN based neural networks. Application of visual stimulus in RAIN neural networks is also a pending project on the lines of audio stimulus project. Projects exploring possibility of applying RAIN principle to model and simulate memory, learning, consciousness and various other human principles are an exciting prospect.

The benchmark tests revealed that there is a scope for either improving or re implementing the Message Bus with new principles. One feels there are still some small optimizations that could be done to NCS 5, so as to better suit the current hardware configuration. The prospect of adding new features, like XML support, is still open.

Another interesting project prospect would occur if multi-core PCs were bought. Current version of NCS 5 simply creates multiple copies of itself to execute on different nodes. This is actually inefficient in terms of system resources like memory. An alternative suggested requires significant changes to the NCS. The new model would

involve one main copy of the Message Bus. This copy would co-ordinate message flow across nodes and the nodes themselves will only have light-weight NCS process consisting of code required to update cell group dynamics. Messages between these light-weight NCS processes would travel via shared memory implemented using the local memory of that node. Thus multi-core architecture resources would be utilized optimally. This offers a very good solution for expanding the performance efficiency of NCS.

Bibliography

[1] A representation of a simple 3-layer feed-forward ANN with 4 inputs, 5 hidden nodes, and 1 output. [last accessed on November 15th, 2007]; Available from: http://smig.usgs.gov/SMIG/features_0902/tualatin_ann.fig3.gif.

[2] A Synapse. [last accessed on November 15th, 2007]; Available from: http://webspace.ship.edu/cgboer/synapse.gif.

[3] A typical Neuron. [last accessed on November 15th, 2007]; Available from: http://training.seer.cancer.gov/module_anatomy/unit5_2_nerve_tissue.html.

[4] Action Potential. [last accessed on November 15th, 2007]; Available from: http://www.answers.com/topic/action-potential-vert-png.

[5] Alpha and Beta sleep waves. [last accessed on November 15th, 2007]; Available from: http://neurocog.psy.tufts.edu/images/beta_alpha.gif.

[6] Brain Facts and Figures. [last accessed on November 15th, 2007]; Available from: http://faculty.washington.edu/chudler/facts.html.

[7] Cortex Status webpage. [last accessed on November 15th, 2007]; Available from: http://cortex.cse.unr.edu:8000/ganglia/.

[8] Electroencephalography (EEG). [last accessed on November 15th, 2007]; Available from: http://core.ecu.edu/psyc/grahamr/DW_3311Site/BookSections/TextbookSections/Sect1.10.html.

[9] GENESIS, neural network simulator. [last accessed on November 15th, 2007]; Available from: http://www.genesis-sim.org/GENESIS/.

[10] Brain Computation Lab website. [last accessed on November 15th, 2007]; Available from: http://brain.cs.unr.edu/index.php.

[11] Lightfleet interconnect. [last accessed on November 15th, 2007]; Available from: http://www.lightfleet.com/.

[12] NeoCortical Simulator source. [last accessed on November 15th, 2007]; Available from: http://brain.cs.unr.edu/ncsDocs/ncs5.tgz.

[13] NEURON, neural network simulator. [last accessed on November 15th, 2007]; Available from: http://www.neuron.yale.edu/neuron/.

[14] NEUROPLOT: Software for Analysis of Large-Scale NN Data. [last accessed on November 15th, 2007]; Available from: http://brain.cs.unr.edu/publications/neuroplot.m.

[15] On-going electrical activity of the brain. [last accessed on November 15th, 2007]; Available from: http://neurocog.psy.tufts.edu/images/erp.htm.

[16] The Python programming language. [last accessed on November 15th, 2007]; Available from: http://www.python.org.

[17] URBI for Webots documentation. [last accessed on November 15th, 2007]; Available from: http://www.gostai.com/doc/en/webots/.

[18] User Guide for Virtual Social Robot. [last accessed on November 15th, 2007]; Available from: http://brain.cs.unr.edu/share/VSR/doc/VSR_guide.htm.

[19] Webots. [last accessed on November 15th, 2007]; Available from: http://www.cyberbotics.com/.

[20] H. Abdi, D. Valentin, and B. E. Edelman, "Neural Networks", Thousand Oaks: Sage, 1999.

[21] J. L. Blake, and P. H. Goodman, "Speech perception simulated in a biologically realistic model of auditory neocortex", *Journal of Investigative Medicine*, 2004.

[22] J. M. Bower, and D. Beeman, "The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System", Springer, 1998.

[23] M. A. B. Brazier, "The Electrical Activity of the Nervous System", written at London, Pitman, 1970.

[24] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris, Jr., M. Zirpe, T. Natschläger, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. E. Boustani, A. Destexhe, "Simulation of networks of spiking neurons: A review of tools and strategies", *J. Comput. Neuroscience*, vol. 23, pp. 349-398, 2007.

[25] D. Crevier, "AI: The Tumultuous Search for Artificial Intelligence", New York, NY: BasicBooks, 1993.

[26] A. Destexhe, and E. Marder, "Plasticity in single neuron and circuit computations", *Nature*, vol. 431, pp. 14, 2004.

[27] R. Doursat, and P. H. Goodman, "Neocortical Keys and Locks: A Neural Model of Associative Learning by Coherence Induction Between Spike Patterns and Ongoing Membrane Potentials", Abstract, Submitted 2006.

[28] D. Drachman, "Do we have brain to spare?", *Neurology*, vol. 64(12), 2005.

[29] R. Drewes, "Brainlab: a toolkit to aid in the design, simulation, and analysis of spiking neural networks with the NCS environment", [Master's Thesis], University of Nevada, Reno, 2005.

[30] R. Drewes, J. Maciokas, S. J. Louis, and P. H. Goodman, "An evolutionary autonomous agent with visual cortex and recurrent spiking columnar neural network", In *Proceedings of the 2004 Genetic and Evolutionary Computing Conference (GECCO 2004)*, vol. 3, pp. 257:258, 2004.

[31] J. Duncan, and A. M. Owen, "Common regions of the human frontal lobe recruited by diverse cognitive demands", *Trends Neuroscience*, vol. 23, pp. 475-483, 2000.

[32] J. Frye. "Parallel Optimization of a NeoCortical Simulation Program", [Master's Thesis], University of Nevada, Reno, December 2003.

[33] J. Frye, J. G. King, J. C. Wilson, and F. C. Harris, Jr., "QQ: Nanoscale Timing and Profiling", in the *Proceeding of the 19th IEEE International Parallel & Distributed Processing Symposium*, 2005.

[34] P. H. Goodman, S. Buntha, Q. Zou, and S. Dascalu, "Virtual neurorobotics (VNR) to accelerate development of plausible neuromorphic brain architectures", *Frontiers in Neurorobotics*, 2007.

[35] P. H. Goodman, R. Doursat, Q. Zou, M. Zirpe, and O. Sessions, "RAIN Brains: Mammalian Neocortex as a Hybrid Analog-Digital Computer", *Poster at Unconventional Computation Conference*, Santa Fe, March 2007.

[36] P. H. Goodman, and F. C. Harris, Jr., "Parallel Beowulf Brain-Robotics Simulation", Research Proposal to ONR, Submitted 2001.

[37] P. H. Goodman, and F. C. Harris, Jr., "Durip04: Parallel Beowulf Computing/Brain/Robotics, Phase III", Research Proposal to ONR, Submitted 2004.

[38] P. H. Goodman, S. J. Louis, and H. Markram, "Parallel Beowulf Brain Simulation", Research Proposal to ONR, Submitted 1999.

[39] P. H. Goodman, E. C. Wilson, J. B. Maciokas, F. C. Harris, Jr., S. J. Louis, A. Gupta, and H. J. Markram, "Large-scale parallel simulation of physiologically realistic multicolumn sensory cortex", Tech Report 01-01, 2001.

[40] J. R. Gray, T. S. Braver, and M. E. Raichle, "Integration of emotion and cognition in the lateral prefrontal cortex", *Proceedings of the National Academy of sciences of the USA*, vol. 99, pp. 4115-4120, 2002.

[41] D. Hanselman, and B. Littlefield, "Mastering Matlab 7", Pearson Prentice Hall, 2005.

[42] D. O. Hebb, "The organization of behavior", Wiley, 1949.

[43] C. Johansson, and A. Lansner, "Towards cortex sized artificial neural systems", *Neural Networks*, vol. 20(1), pp. 48–61, 2007.

[44] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, "Principles of Neural Science", 4th ed. McGraw-Hill, New York, 2000.

[45] J. G. King, "Brain Communication Server: A Dynamic Data Transferal System for A Parallel Brain Simulator", [Master's Thesis], University of Nevada, Reno, 2005.

[46] J. C. Macera, "Design and Implementation of a Hierarchical Robotic System: A Platform for Artificial Intelligence Investigation", [Master's Thesis], University of Nevada, Reno, 2003.

[47] J. C. Macera, P. H. Goodman, F. C. Harris, Jr., R. Drewes, and J. Maciokas, "Remote-neocortex control of robotic search and threat identification", *Robotics and Autonomous Systems*, vol. 46(2), pp. 97-110, 2004.

[48] J. B. Maciokas, "Towards an Understanding of the Synergistic Properties of Cortical Processing: A Neuronal Computational Modeling Approach", [PhD Thesis], University of Nevada, Reno, 2003.

[49] J. B. Maciokas, P. H. Goodman, and F. C. Harris, Jr., "Large-scale spike-timing-dependant-plasticity model of bimodal (audio-visual) processing", Technical Paper, Brain Computation Lab, University of Nevada, Reno, 2002.

[50] J. B. Maciokas, P. H. Goodman, and J. L. Kenyon, "Accurate Dynamical Model of Interneuronal GABAergic Channel Physiologies", Technical Paper, University of Nevada, Reno, 2004.

[51] P. Marchand, and O. T. Holland, "Graphics and GUIs with Matlab", Chapman & Hall/CRC, 3rd ed., 2003.

[52] H. Markram, P. Dimitri, A. Gupta, and M. Tsodyks, "Potential for multiple mechanisms, phenomena and algorithms for synaptic plasticity at single synapses", *Neuropharmacology*, vol. 37, pp. 489–500, 1998.

[53] H. Markram, J. Lubke, M. Frotscher, A. Roth, and B. Sakmann, "Physiology and anatomy of synaptic connections between thick tufted pyramidal neurons in the developing rat neocortex", *J. Physiology*, vol. 500, pp. 409-440, 1997.

[54] H. Markram, J. Lubke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs", *Science*, vol. 275, pp. 213-215, 1997.

[55] R. S. Michalski, and George Tecuci, "Machine Learning: A Multistrategy Approach", Volume IV, Morgan Kaufmann, 1994.

[56] V. Mountcastle, "The columnar organization of the neocortex", *Brain*, vol. 120, pp. 701-722, 1997.

[57] B. Opitz, "A Balanced Knock-Out Computer Model of Neuronal Ca++-Dependant K+ Channels", Technical Paper, University of Nevada, Reno, 2004.

[58] Q. Peng, "Brainstem: A NeoCortical Simulator Interface for Robotic Studies", [Master's Thesis], University of Nevada, Reno, December 2006.

[59] D. Poole, A. Mackworth, and R. Goebel, "Computational Intelligence: A Logical Approach", Oxford University Press, 1998.

[60] D. Purves, G. J. Augustine, D. Fitzpatrick, W. C. Hall, A. S. LaMantia, J. O. McNamara, and S. M. Williams, "Neuroscience", 3rd ed., Sinauer Associates Inc., 2004.

[61] M. C. Ripplinger, C. J. Wilson, J. G. King, J. Frye, R. Drewes, F. C. Harris, Jr., and P. H. Goodman, "Computational model of interacting brain networks", *Journal Of Investigative Medicine*, vol. 52, pp. S155-S155, 2004.

[62] S. J. Russell, and P. Norvig, "Artificial Intelligence: A Modern Approach", 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2003.

[63] The Editors of Scientific American, "The Scientific American Book of the Brain", New York: Scientific American, pp. 3, 1999.

[64] G. Tononi, and C. Cirelli, "Sleep function and synaptic homeostasis", *Sleep Medicine Reviews*, vol. 10, pp. 49–62, 2006.

[65] M. V. Tsodyks, and H. Markram, "The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability", *Proc. Natl. Acad. Sci.*, vol. 94, pp. 719-723, 1997.

[66] K. Tsunoda, Y. Yamane, M. Nishizaki, and M. Tanifuji, "Complex objects are represented in macaque inferotemporal cortex by the combination of feature columns", *Nature Neuroscience*, vol. 4(8), pp. 832–838, 2001.

[67] B. D. Ventura, C. Lemerle, K. Michalodimitrakis, and L. Serrano, "From *in vivo* to *in silico* biology and back", *Nature*, vol. 443, pp. 5, 2006.

[68] T. P. Vogels, and L. F. Abbott, "Signal Propagation and Logic Gating in Networks of Integrate-and-Fire Neurons", *The Journal of Neuroscience*, vol. 25(46), pp. 10786–10795, 2005.

[69] K. K. Waikul, L. Jiang, F. C. Harris, Jr., and P. H. Goodman, "Implementation of a web portal for a neocortical simulator", In *Proceedings of CATA*, 2002.

[70] Y. Wang, H. Markram, P. H. Goodman, T. K. Berger, J. Ma, and P. S. Goldman-Rakic, "Heterogeneity in the pyramidal network of the medial prefrontal cortex", *Nature Neuroscience*, vol. 9, pp. 534–542, 2006.

[71] R. W. Williams, and K. Herrup, "The Control of Neuron Number", *The Annual Review of Neuroscience*, vol. 11, pp. 423–453, 1988.

[72] E. C. Wilson, "Parallel implementation of a large-scale biologically realistic parallel neocortical-neural network simulator", [Master's Thesis]. University of Nevada, Reno, 2001.

[73] E. C. Wilson, P. H. Goodman, and F. C. Harris, Jr., "A Large-Scale Biologically Realistic Cortical Simulator", in the *Proceedings of SC 2001*, Denver, Colorado, 2001.

[74] E. C. Wilson, P. H. Goodman, and F. C. Harris, Jr., "Implementation of a Biologically Realistic Parallel Neocortical-Neural Network Simulator", in the *Proc. of the 10th SIAM conference on Parallel Process for Sci. Comput.*, 2001.

[75] D.G. Bobrow, S. Mittal, and M.J. Stefik, "Expert systems: perils and promise", *Communications of the ACM*, vol. 29(9), pp. 880-894, 1986.

[76] D. Hinkle, and C.N. Toomey, "CLAVIER: Applying case-based reasoning on to composite part fabrication", *Proceeding of the Sixth Innovative Application of AI Conference, Seattle, WA, AAAI Press*, pp. 55-62, 1994.

[77] G.A. Davis, "Bayesian reconstruction of traffic accidents", *Law, Probability and Risk*, vol. 2, pp. 69-89, 2003.

[78] J.B. Kadane, and D.A. Schum, "A Probabilistic Analysis of the Sacco and Vanzetti Evidence", *Wiley, New York*, 1996.

[79] R. A. Brooks, "Cambrian Intelligence", *MIT Press*, 1999.

[80] R.C. Arkin, "Behavior-Based Robotics", *MIT Press*, 1998.

[81] R.D. Beer, R.E. Ritzmann, and T.M. McKenna, "Biological neural networks in invertebrate neuroethology and robotics", *Academic Press, Boston*, 1993.

[82] NCS User Documentation [last accessed on November 15th, 2007]; Available from: http://brain.cs.unr.edu/ncsDocs/ncsUser/TOC.html.

[83] G. Cybenko, "Approximation by Superpositions of a Sigmoidal Function", *Math. Control, Signals Sys.*, vol. 2, pp. 303, 1989.

[84] Z. Gu, L.H. Lam, and P.S. Dhurjati, "Feature correlation method for enhancing fermentation development: A comparison of quadratic regression with artificial neural networks", *Computers & Chemical Engineering*, vol. 20, pp. S407-S412, 1996.

[85] H.T. Siegelmann, "Foundations of Recur rent Neural Networks", [PhD dissertation], New Brunswick Rutgers, The State University of New Jersey, 1993.